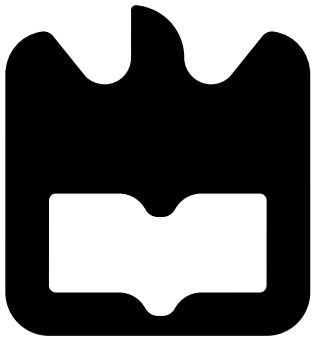
Universidade de Aveiro Departamento de Electrónica, Telecomunicações e Informática,
2017

Filipe Daniel Azevedo Martins

YouInteract: Uma aplicação para interação por gestos com ecrãs públicos

YouInteract: An application for gesture interaction with public displays



Filipe Daniel Azevedo Martins

YouInteract: Uma aplicação para interação por gestos com ecrãs públicos

YouInteract: An application for gesture interaction with public displays

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Paulo Miguel de Jesus Dias, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Manuel Leite da Silva, *software developer* na Fábrica Centro Ciência Viva de Aveiro .

o júri / the jury

presidente / president

Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Frutuoso Gomes Mendes da Silva
Professor Auxiliar do Dep. de Informática da Universidade da Beira Interior

Professor Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar da Universidade de Aveiro

agradecimentos

É com muito gosto que aproveito esta oportunidade para agradecer ao Professor Paulo Dias e ao Doutor João Silva, orientadores da dissertação, por todo o apoio, disponibilidade e contínuo acompanhamento durante a realização deste trabalho. Agradeço também à Fábrica Centro Ciência Viva de Aveiro por ser parceira neste projecto e particularmente ao Jorge Godinho por disponibilizar o seu tempo e todos os recursos possíveis para a concretização deste trabalho.

Resumo

Com o lançamento de equipamentos para reconhecimento de gestos a preços acessíveis assim como o aparecimento e baixa de preços de ecrãs de grandes dimensões que tornam possível a sua instalação em diferentes locais, é agora interessante criar um sistema interativo que possa ser instalado em diversos espaços públicos com a finalidade das pessoas poderem interagir com conteúdo relevante. Esta dissertação continua um projecto desse tipo, que visa permitir a interação através de gestos em locais públicos, existente no Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro. O objectivo é tornar o sistema mais abrangente, permitindo criar vários conteúdos e configurar os mesmos através de um website. Apesar da aplicação poder ser configurada/usada em vários contextos, foi aproveitada uma parceria com a Fábrica Centro Ciência Viva de Aveiro para realizar dois testes pilotos com utilizadores, permitindo que se pudesse avaliar a solução implementada e a variedade de aplicações desenvolvidas. Esta dissertação apresenta os resultados deste trabalho que visa uma maior exploração de dispositivos interativos em espaços públicos.

Abstract

With the release of affordable gesture-tracking hardware and the launch on the market of cheaper large displays making it possible to install them in various locations, it is now interesting to create an interactive system to be installed in public spaces to allow users to interact with relevant information. This dissertation continues such a system project developed at Department of Electronics, Telecommunications and Informatics of University of Aveiro. The objective is to further improve and develop a system that allows the creation of additional content and its configuration through a website. Although the final application might be configured for different context, we developed this work in collaboration with Fábrica Centro Ciência Viva de Aveiro to perform two pilot studies, evaluating the designed solution and the acceptance of the new created applications. This dissertation presents the results of this work hoping to contribute to the use of gesture interaction with public displays.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Motivation

The developments in low-cost tracking systems, such as Microsoft Kinect and gesture recognition algorithms led to the increasing popularity of gesture-based interfaces [Garber, 2013]. In addition to this, the massification of thinner and cheaper displays that can be installed in public spaces like lobbies, shops, coffee houses, museums and many others, opened an affordable opportunity. By joining these two technologies it is possible to create a system that shows different types of information to users while also offering the user a chance to interact with it. This work focuses on creating gesture based interactive displays that would allow to easily upgrade an existing static public display to an interactive one. With a partnership with Fábrica Centro Ciência Viva de Aveiro (FCCVA), it could be possible to create a robust and interesting system for public displays that could be different of the existing ones on the market. Instead of always showing the same content and applications without a chance of updating them, a new system that could be able to be configured according to the needs and to integrate new future applications would be an important evolution of gesture interaction systems with public displays.

1.2 Objectives

This dissertation intends to improve and integrate in a single system several graduate and master level projects [Palha, 2012], [Cardoso, 2015], [Duarte, 2011], [Parracho, 2013] developed through the last years regarding interaction with public displays. The final goal is a system able to adapt to multiple environments so a new architecture must be developed and evolved according to the results from the tests made during the time of this dissertation. Another objective is the development of an API that will allow the development of further applications to integrate in this system. To conclude, this dissertation seeks to have a final application which uses gesture recognition in order to provide an interaction system-user, that can be configured through a web site and that has interesting content for people to interact with it.

1.3 Structure

This dissertation is divided into seven main chapters. The first chapter introduces the motivation and objectives behind this work. Chapter two begins by giving an overview of related work and the pros and cons observed in each one of them and ends with an analysis of previous versions of the application developed in this dissertation. The third chapter describes the architecture and tools that were used in order to adapt and improve the existing web portal used to support the main application. Chapter four describes the core application and gives a detailed explanation on how the final version was reached and the steps taken during the dissertation time in order to reach it. The fifth chapter is concerned about a specific part of the main application regarding the integration of several applications. Chapter six is concerned with user studies and describes the methodology followed by the analysis and discussion of the obtained results in the two days of tests at FCCVA. Finally, chapter seven concludes this dissertation by presenting the main results, the limitations of the developed work and implementation challenges, as well as suggesting possible improvements and future work.

Chapter 2

Related Work

This chapter provides an overview of works focused on the interaction between users and public displays. This analysis will be divided in two main cases: public displays using Microsoft Kinect for gesture interaction with the system and a public display system developed by several students of University of Aveiro.

2.1 Public Displays with Kinect

Bowman, in the book "3D User Interfaces" [Bowman, 2014], points 3D user interfaces as the natural choice for large display contexts. Bowman also stats in [Bowman et al., 2004] that traditional mouse and keyboard are difficult to use because public displays are meant to call to interaction multiple users. So an hardware capable of tracking users' gestures can be a good solution to offer an user-system interaction.

Microsoft Kinect offer an affordable way for gesture-tracking. It is being used in multiple areas such as education [Hsu, 2011] and [Boutsika, 2013], physical rehabilitation [Cassola et al., 2014]. The papper [Zhang, 2012] says "Kinect's impact has extended far beyond the gaming industry. With its wide availability and low cost, many researchers and practitioners in computer science, electronic engineering, and robotics are leveraging the sensing technology to develop creative new ways to interact with machines and to perform other tasks".

The paper by [Motta and Nedel, 2013] explores the interaction between a user and a public display using an extended version of Kinect SDK. It refers many existing works with natural user interfaces deal with interactions using touchscreens, mobile devices and even data gloves, devices that require extensive setups so they are not suitable for public displays. Microsoft Kinect came as a low cost device that could recognize human gestures with relatively good results (Figure 2.1).

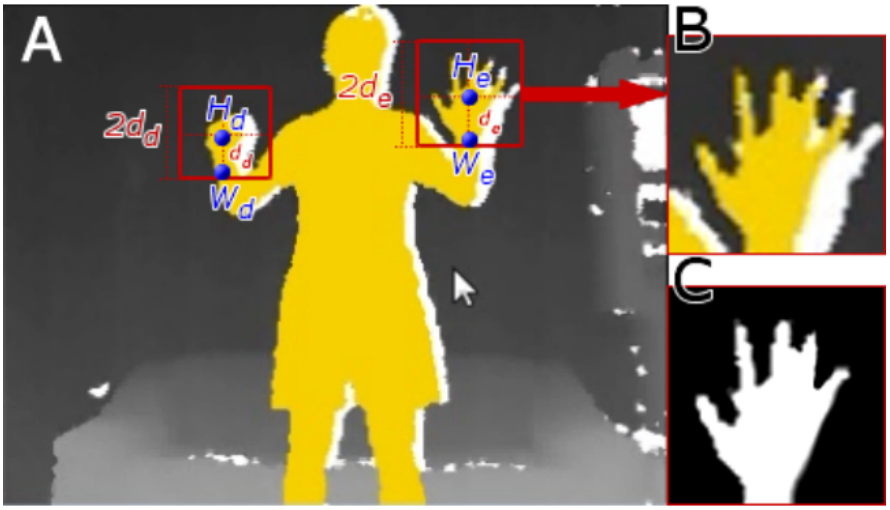


Figure 2.1: Hand detection by Microsoft Kinect SDK and its depth segmentation in order to detect the opening and closing of the hand, [Motta and Nedel, 2013].

With the recognized gestures available in Kinect SDK and this new implemented algorithm, a user experiment was conducted in order to verify how easy it was to select and manipulate simple virtual objects. Several squares with different sizes were presented to the user and the objective was to select every one of them in a specific order and then drag the last square, by closing the hand, to a certain screen locations (Figure 2.2).

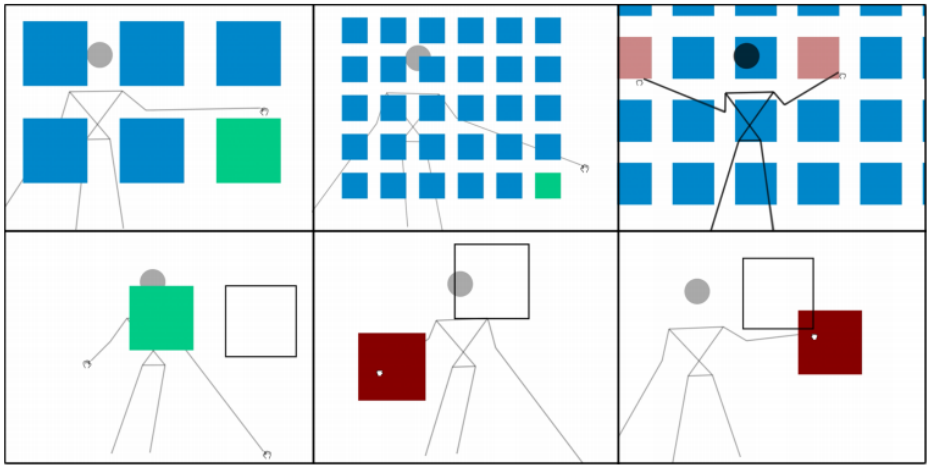


Figure 2.2: Tasks given to the users consisted on selecting the squares with different sizes in a specific order and then drag the last one to a shown location, [Motta and Nedel, 2013].

With the setup installed as Figure 2.3 indicates, the tests were made with the help of people passing by that location. With this experiment, the authors concluded that the recognition available at Kinect SDK although not behaving very accurately in all situations such as selecting objects with small sizes, is good enough for most of the public displays setups as well as their algorithm to recognize the opening and closing of the hand also functioned satisfactorily.

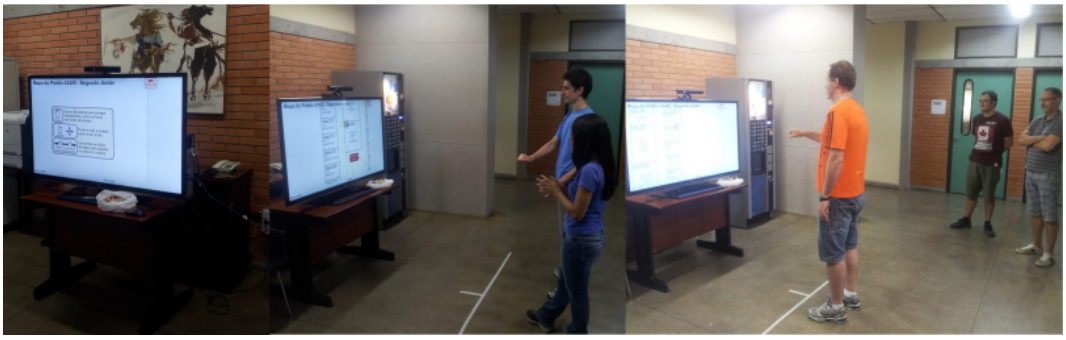


Figure 2.3: Location of where the setup for the user experiment was installed and how each user interacted with the system, [Motta and Nedel, 2013].

The paper by [Walter et al., 2014] refers also to a system that recognizes mid-air gestures in order to provide an interaction between the user and the system (Figure 2.4).

Figure 2.4: Microsoft Kinects used to capture users movements for interaction, [Walter et al., 2014].

In order to know the best mechanisms for people to select specific data shown, such as images, text and buttons, they first analyzed the user behaviour in a pilot study by asking the users to select a red item without any other instructions (Figure 2.5). The results showed that most of the users were able to reach the target spot and in order to select it they would leave their hand above the red item, a gesture called Point+Dwell. After adjusting the selection techniques in an iterative design study, the usability of the system was improved and a new field study in three different locations was conducted.

Figure 2.5: The participant was asked to select the red item without any other instruction, [Walter et al., 2014].

This final study was done in three different public spaces where people participated individually or in groups (Figure 2.6). By analyzing the users' behaviour and gestures, the conclusion was that without any hint provided, people were successful in selecting items. In addition to this, most people tend to explore multiple items before selecting one. It also was possible to see that in a public context, people interact inadvertently with the system while doing other things.

Figure 2.6: Multiple people interacting with the system, [Walter et al., 2014].

2.2 YouInteract's Journey

This dissertation continues previous work , [Palha, 2012], [Cardoso, 2015], [Duarte, 2011], [Parracho, 2013] on a public display system designed and created by students of the University of Aveiro's Department of Electronics, Telecommunications and Informatics (DETI), who had the idea of creating a system that showed relevant information of the department.

The project started in 2009 [Dias et al., 2014] with the installation of displays in DETI's lobby and the interaction with it being made by an Android smartphone with a specific application installed. The communication between the smartphone and the application was done using Bluetooth and, in this version, the user had the possibility to navigate through webpages based on the department website. This navigation was done by two methods. The first one used the phone's accelerometer where users navigated through the pages by turning the mobile phone up, down, left or right (Figure 2.7). The other method was to use mobile phone's touchscreen as a computer touchpad to control a virtual pointer that appeared in the display.

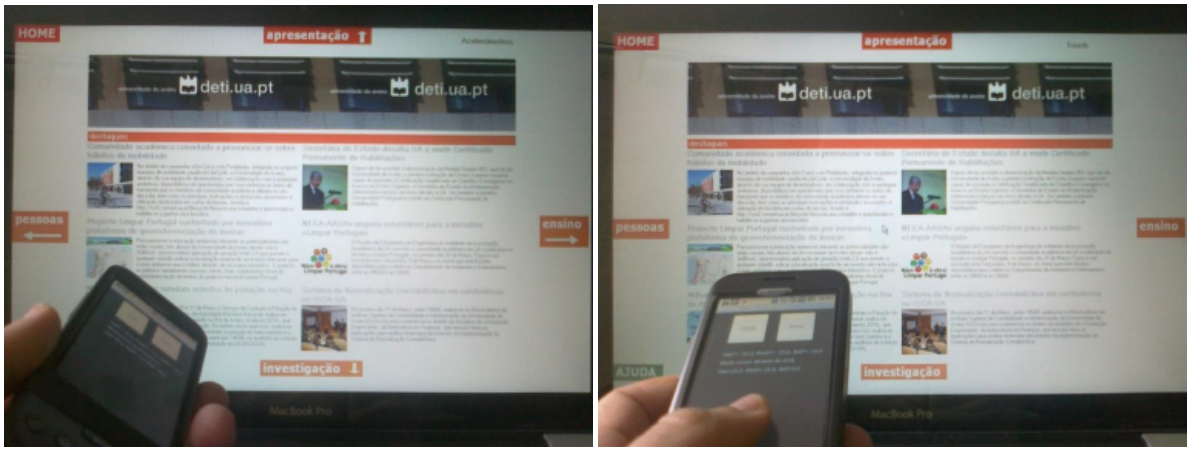


Figure 2.7: On the left it is shown how by moving the phone it changed the pages and on the right it is shown that by clicking in the phone's touchscreen it would select the page.

The system evolved in 2011-2012 with a master's thesis [Palha, 2012] and, instead of displaying static pages like the older version, it showed information stored in a department's server such as course schedules and faculty members' contact information which was important content to the students. In addition to this, new ways of interacting with the system were implemented such scroll navigation, fling to change the selected item, tap to confirm the selected item, long press to enter and exit specific interaction modes and rotate the device to change pages (Figure 2.8). Although the system displayed content that students regularly needed, it was hardly used because it demanded the user a lot of work by downloading the application, enabling the Bluetooth adapter and finally pair it with the system computer.

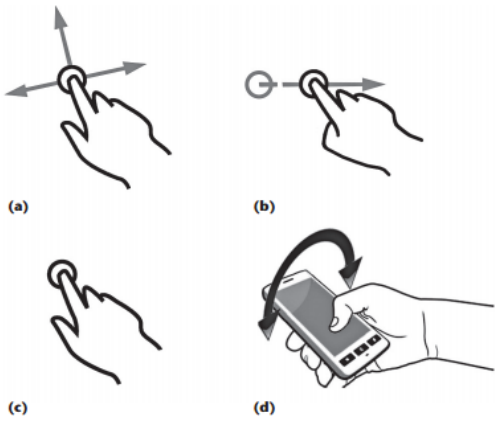


Figure 2.8: Movements that made possible the interaction with the system. (a) Scroll, (b) Fling, (c) Tap or long press, and (d) Rotate.

To solve the problem of having the need to pair the device with the system computer, and since Microsoft Kinect had just been release in November 2010, the system was adapted [Parracho, 2013] in order to work with full body gesture instead of mobile phone interaction (Figure 2.9). The gestures recognized were the ones available at the Microsoft SDK such as pull and push and others like swipe or two-hand gestures - which later was removed since the users had difficulties performing it. The next version, developed over 2012-2013 , introduced a menu with icons of the applications available and a virtual avatar that would replicate the user movements in order to call for his attention to use the system.

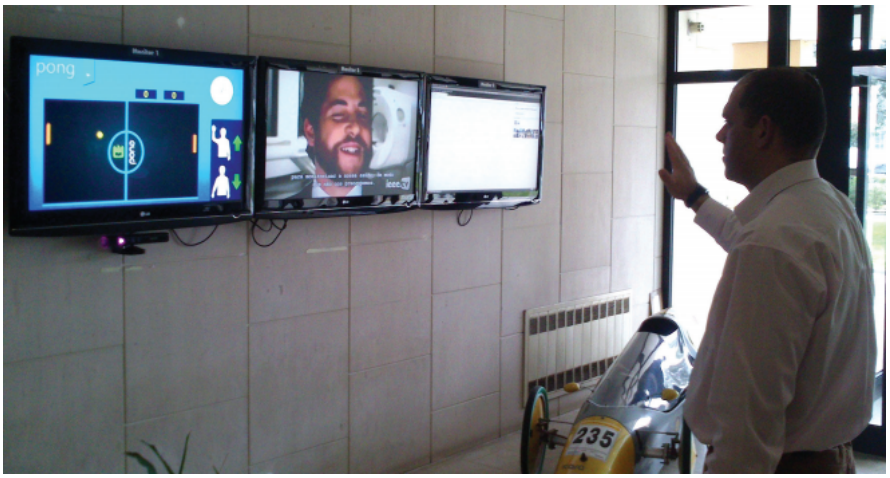


Figure 2.9: Users navigate through the system using their hands.

In 2014-2015, the last version of DETI-WALL before this dissertation [Cardoso, 2015], in addition to some new applications such as a Tic-Tac-Toe game and an application that allowed to see videos, it was developed a PHP website in order to provide the system administrator the possibility to configure YouInteract via Internet. In this portal, the administrator was able to upload new applications (Figure 2.10) and modify some parameters of its configuration. This portal was a big step in the project since it allowed the system administrator to upload new content without having the need to access physically the computer.

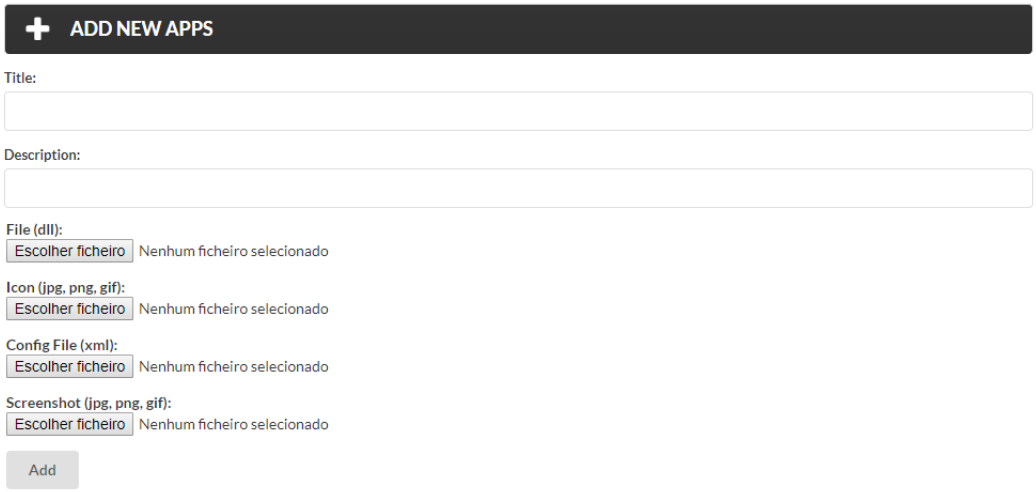


Figure 2.10: Example of a Portal's webpage where the administrator could upload a new application.

2.3 YouInteract proposal

After analyzing the last version of DETI-Wall it was concluded that it was needed to make major changes in the architecture of the system in order to accomplish the goals proposed. Since the project had been in so many different hands along the years and each group of students developed new features without knowing very well what was done before and why it was that way, the whole architecture became just a sum of pieces of blocks, without a plan on what to do next. With this thought, it was decided the new version would need a new architecture although it was possible to take advantage of some of the blocks already created.

The next chapters describe the work done in order to have a functional YouInteract version with several new features that make this version not only an evolution from the previous work but also a disruption since it is now a system that can be adapted to be used in diverse public display locations.

Figure 2.11 shows a macro architecture proposal for this dissertation. It can be divided in three major blocks that will be detailed explained in the next chapters. It starts by explaining the YouInteract Portal, a website that is a fundamental tool to configure and personalize YouInteract with the desired content (Chapter 3). Next, it is explained the YouInteract Core which is the main application that reads the configuration files provided by the Portal and integrates several other applications (Chapter 4). It is explained the type of applications that can be integrated in the YouInteract system, how it is done and several examples of applications developed until the end of this dissertation (Chapter 5), the user tests made to verify the solution created (Chapter 6) and the conclusions about the work and results of the tests (Chapter 7).

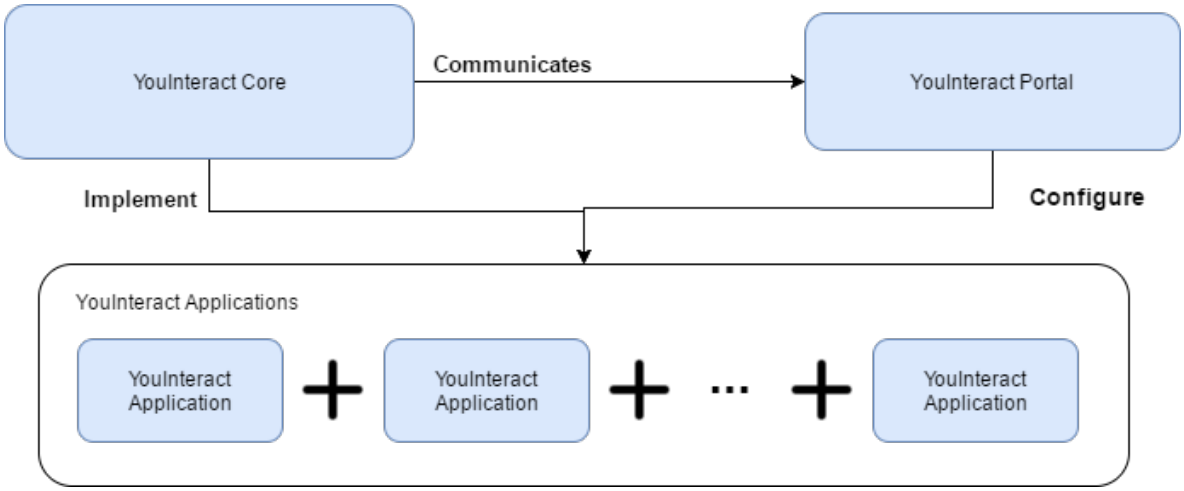


Figure 2.11: YouInteract's system main components

Chapter 3

YouInteract:Portal

YouInteract is a system designed to adapt to different types of configuration. The idea is to allow an administrator to personalize several features of the system which will change its behaviour and interface (see Appendix A for how to locally install the Portal in the computer). These changes are made through using a web Portal designed to give a full control of setup configurations. The next sections will describe how YouInteract's Portal is designed, its main features and how it will communicate with the computers which have YouInteract installed and the Kinect sensor.

As described before, the Portal designed and created for the previous versions of YouInteract was exhaustively tested and revealed to be stable, robust and its interface well organized. Despite this, it was made an analysis of the technologies used and compared to the existing ones nowadays in order to assure the Portal was created using the new and best versions of the technologies available today in order for it to be the most robust possible.

With this in mind, it was decided to not start from zero by creating a new portal and continue the good work taking advantage of an already existing structure. Over the next sections, but mainly in section 3.4, it is explained the final version of YouInteract Portal. It has the same structure has the version before this dissertation, however it was fixed some bugs, added new features, changed the user interface in some parts of the website as well as changed the database. This lead to the removal, change and addition of files related with the website's Models, Views, Controllers and Database (explained in sections 3.4.2, 3.4.3, 3.4.4 and 3.4.5). All of these changes were made to ensure a good behaviour in every of the use cases available in the Portal (Figure 3.5).

3.1 Concept

YouInteract Portal is a website that starts by showing every computer running YouInteract and offers a chance to configure each computer's YouInteract in order to behave in a certain way (see Appendix B for a manual of the Portal). This website enables everyone with a registered account to add and modify the available content as well as change the behaviour of YouInteract according to the needs. If it is needed to, every time YouInteract launches, show a specific application, the administrator may choose it through this portal. If it is needed for the computer to hibernate and then awake at specific hours, the administrator can choose it through this portal. If the administrator wants to change certain interface elements such as the background, Menu's icons and button layouts, the portal also supports it.

In addition to all of the configurations, in the website it is also possible to upload new applications, images, videos and other items that will be available in YouInteract. As an example, if the administrator uploads a video through the website, that video will be available in every chosen computer running YouInteract.

As Figure 3.1 shows, YouInteract Portal is a compound of a website and its database, which communicates with a cloud environment where the YouInteract's system is installed (currently Dropbox but it can be changed by the administrator). Depending on the configurations chosen, the portal will create .xml files with information that will be read by YouInteract:Core (Chapter 4).

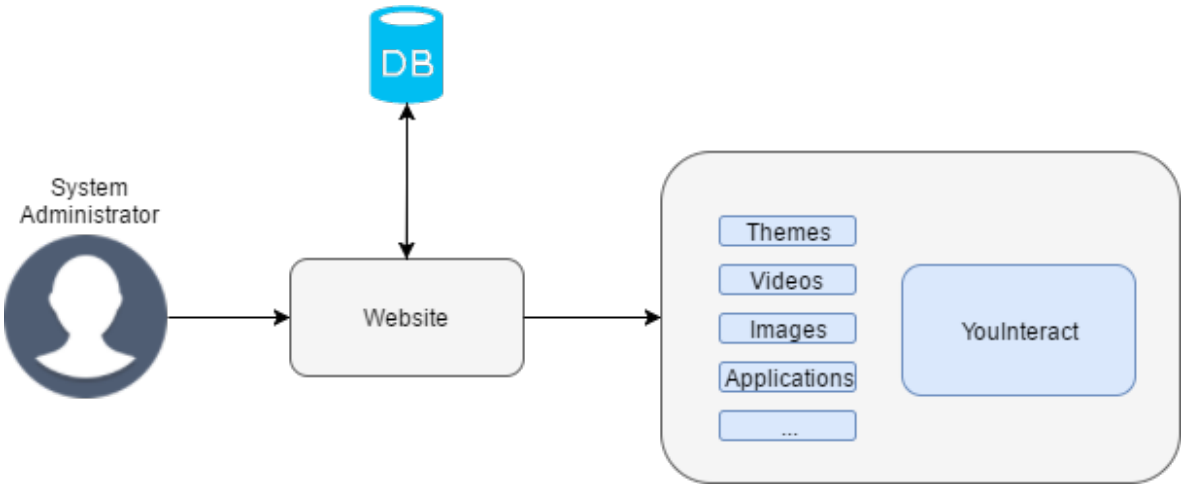


Figure 3.1: Diagram describing how the system is organized and divided.

3.2 Use Cases

There are three types of users within Portal's system: the visitors, the YouInteract's administrators and the Portal administrators (Figure 3.2).

Visitors can only access public information available in the website. Since they are not registered in the system, they can only consult basic information about YouInteract project such as the names of all the people that worked directly in this project, videos explaining and showing what the system is capable of and contact information to reach its developers. In addition to this, visitors may also download the last version of the YouInteract's API in order to develop new applications. The website also gives access to a manual and documentation that help future developers to understand the YouInteract and develop new applications for it.

YouInterat's administrators are able to access to every information related to YouInteract including the computers where YouInteract is installed. Through this website, system administrators can configure the YouInteract application running in a specific device in order to have it behaving the way it suits the needs of the administrator. This Portal offers a number of configurable options that can change the interface and the behaviour of a computer that is running YouInteract.

Portal administrators have all the functions available to YouInteract's administrators but they may also control what is changed in the website. They have a log information about the changes made through the Portal and the user account that made them. This role has also the possibility to change some of the portal's settings such as the default configurations and default templates in addition to settings related to the appearance of the website (devices per page, items per page...). Portal administrators are the only ones who can add more YouInteract's administrators or Portal administrators to the system.

Figure 3.2 shows the most important use cases available in the website for each of the role.

Figure 3.2: Most important use cases available on the Portal.

- Visitor
  - Consult project's description - Any user can check information related to YouInteract such as a background explanation of the project, videos showing how it functions and contact information.
  - Download YouInteract's API - Any user can download the API used to develop new applications for YouInteract.
  - Consult API Documentation - Any user can consult the documentation needed to understand YouInteract's API.
- YouInteract's administrator
  - Check the devices running YouInteract and modify their configurations and data - The administrator can modify data used to identify each device (its name for example) as well as the configuration that each device will have. This configuration includes which applications YouInteract will have, the background template, the system behaviour, its start up application, among others.
  - Check, modify, add and delete configurations - The administrator have full control in the configurations available in the Portal. He can have several types of configurations and use them in the ways he needs.
  - Check, modify, add and delete templates - Each configuration has its own template setting namely the background image the YouInteract will show to the users. Administrators have full control on adding, removing and editing templates.
  - Check, modify, add and delete applications - Administrators can add new applications YouInteract is able to run. After adding one application, it will be available to be selected in a configuration that a device will have.
  - Check, modify, add and delete items - Administrators can upload files such as images, videos and rss feeds that will be available in YouInteract. It is uploaded one by one and the administrator has full control on which name, description and much more, each item will have.
  - Import photos/videos at once - Administrators may upload several images and videos at once, making the job easier when there are numerous items to upload.
- Portal's administrator
  - Update YouInteract's API version available to download - Portal administrators can modify the API file available for visitors to download.
  - Check activiy log - Portal administrators have information about all log ins and changes made in the website.
  - Add and delete administrators - Portal administrators may add new YouInteract or Portal administrators to the system as well as remove some of them.

3.3 Used Technologies

The structure of the existing portal was studied and then the technologies used to it. The following sections will explain what it was used previously and the decisions made in order to have the best possible version of the Portal.

It will begin to describe the chosen architectural pattern to structure the website as well as other most known patterns and then the languages used for building the website and database and finally a well known framework that makes the web development job easier and more robust.

3.3.1 Architectural Pattern: MVC

The previous version of the Portal followed the Model-View-Controller (MVC) pattern. However, it was studied nowadays most well-known patterns in order to ascertain this model could continue to be used in the future or it was needed to fully change the structure of YouInteract Portal.

The three more common patterns are MVC (Figure 3.3), Model–view–presenter pattern (MVP) and Model–View–Viewmodel pattern (MVVM). In MVC, the architecture is composed by three entities:

- Model: Represents all the database information as it manages the behaviour of the application. It responds to requests for information the View needs and to instruction changes made by the Controller.
- View: Represents all the visual side where the user will interact with the application.
- Controller: It functions as a middleman between the Model and the View, where it will interpret the user input by either informing the Model and/or View as appropriate or updating the database with the new values.

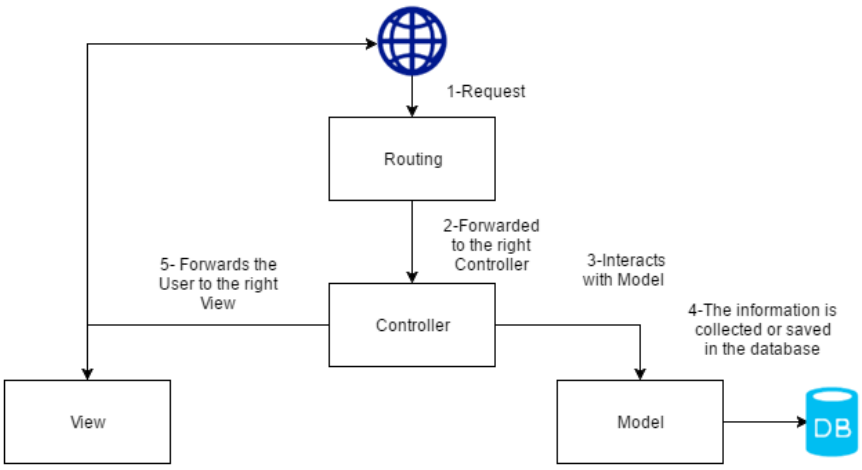


Figure 3.3: Diagram describing how a MVC pattern works.

The other two patterns are derivations of MVC and although having differences, they were not relevant for this type of website. The main difference is that the Controller component

present in the MVC is replaced by the Presenter in MVP and by a ViewModel in MVVM. This affects the communication between Views and Models. Since MVC is one of the most used patterns, with lots of online support and examples, where a great variety of web frameworks available follow it, and also because the previous version of the Portal already used it, it was decided to continue with the MVC pattern.

3.3.2 Laravel Framework

For the Portal's development, [Laravel] framework, which follows the MVC architectural pattern, was used. This is one of the most used frameworks since it helps the developer to produce a more efficient code in a easier and faster way by giving several tools helping the front-end development and the interaction with the database [Laravel-database].

The scripting language used in the framework is [PHP]. PHP is a scripting language designed primarily for web development that can be embedded into HTML. In addition, it is an open source software that supports multiple databases like MySQL or MS-SQL.

For the database [MySQL] was used. MySQL is the most popular relational database management system that guarantees good performance and is easy to use.

3.4 Architecture

This section will describe the architecture of the system. It will begin to explain the system's global structure and then, since the Portal is based on the MVC pattern, it will present information mainly about the Models, Views and Controllers.

3.4.1 Structure

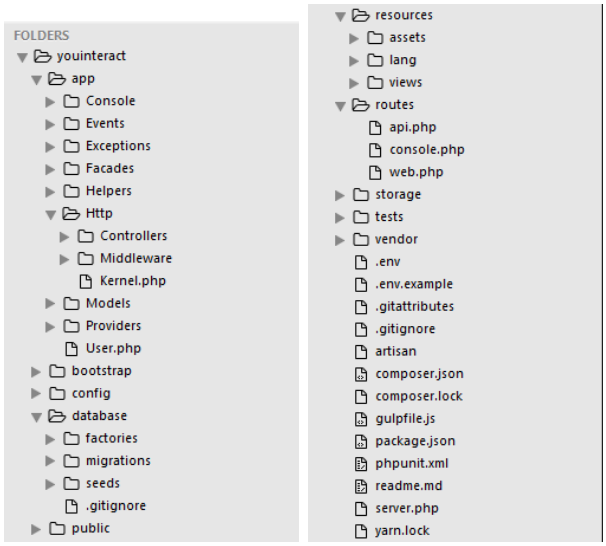


Figure 3.4: Organization of the Portal.

Figure 3.4 identifies the different elements of MVC explained before that together compose YouInteract's structure.

It all starts when the user requests a specific URL (Figure 3.5). The system will locate that same URL in the files responsible for Routing that are present in the "routes" directory of the root and they have all the information to forward every Request to the respective Controller. That Controller will be responsible to interact with the database in order to receive or update data to the table present there. This interaction with the database is done through the Models, present in the app/Models directory. Finally, it needs to show the user the intended information and that is done using the Views, every file related to the visual aspect of the website, which is located is resources/views.

Every file related to the database is present in the database directory of the root where it has a directory for the migrations and other for the seeds.

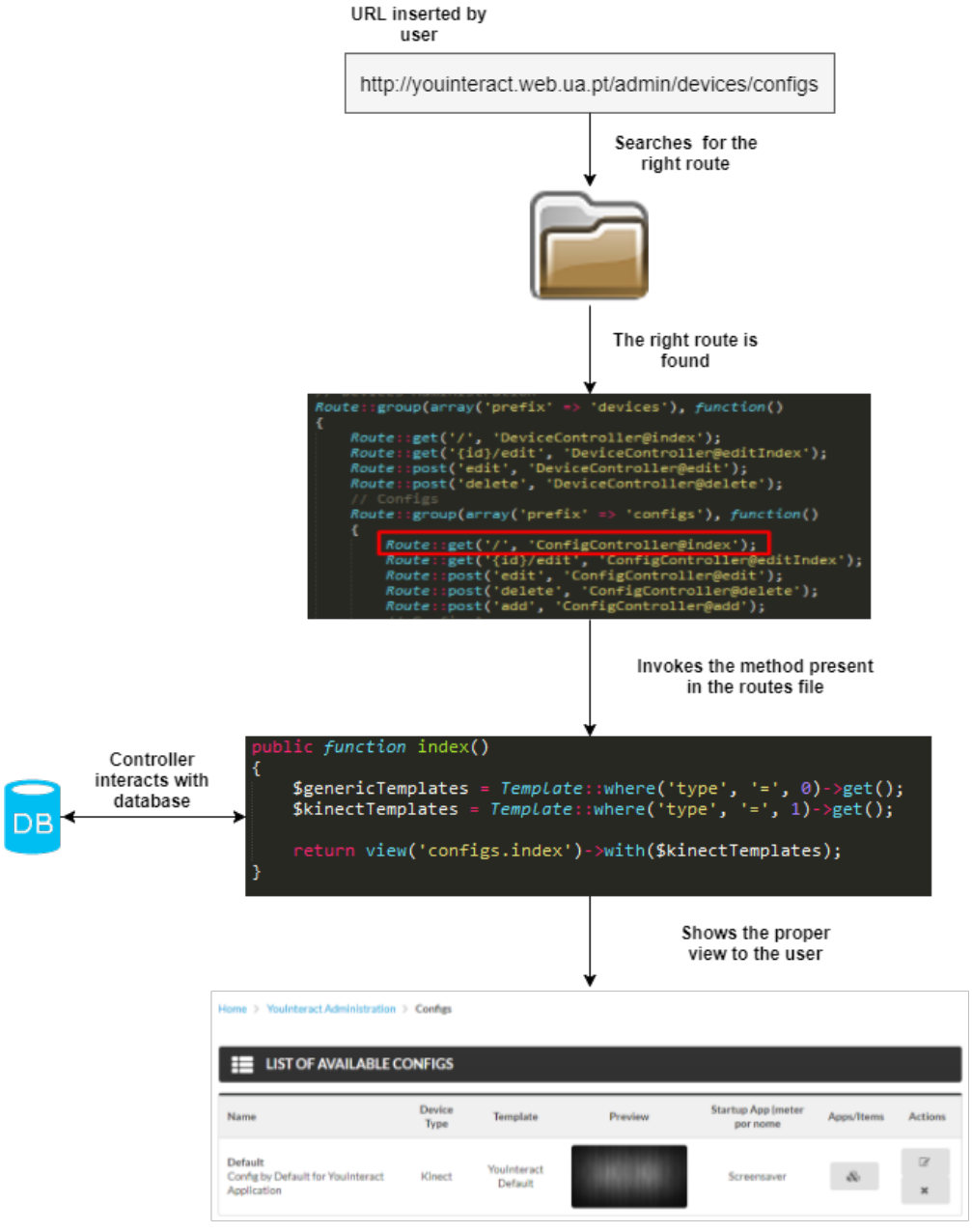


Figure 3.5: Navigation's flow through the Portal.

3.4.2 Models

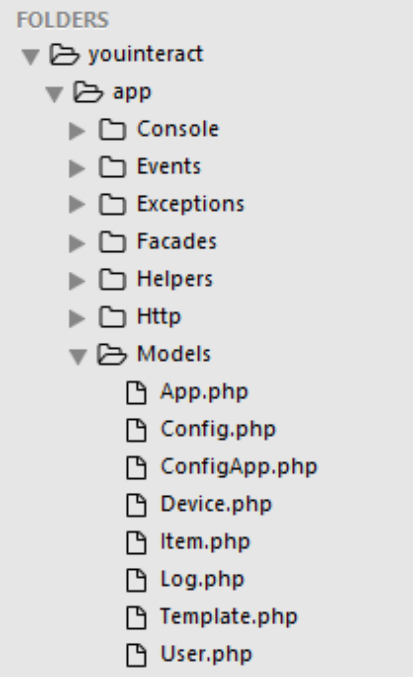


Figure 3.6: Models organization of the Portal.

Figure 3.6 contains all the Models existing in YouInteract. Each model corresponds to a table in the database. It is in the Models files that the relations between every Model is defined as well as methods returning necessary information about that specific model.

- App- Model responsible for the applications.
- Config- Model responsible for the general configuration.
- ConfigApp- Model responsible for the configurations of the applications.
- Device- Model responsible for the devices.
- Item- Model responsible for the available items.
- Log- Model responsible for the log of changes and logins in the website.
- Template- Model responsible for the templates.
- User- Model responsible for the users registered in the website.

3.4.3 Views

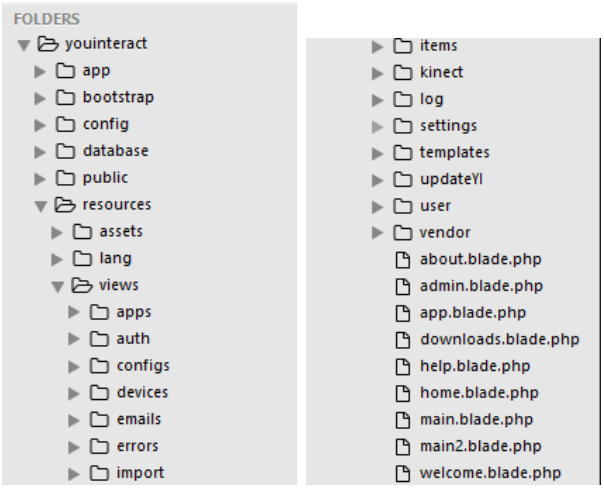


Figure 3.7: Views organization of the Portal.

The views are responsible for the visual section of the system, it is where the user is able to communicate and interact with the website. In order to make the creation of views easier, Laravel framework gives the developer the option to use blade [Laravel-blade], a tool that allows a hierarchy between views. To use it, every file concerning views must end with "blade.php". To take advantage of blade, two main views were created, one for the public section on the website, file main.blade.php in Figure 3.7, and another for the section related to the system administrator, file main2.blade.php in Figure 3.7.

All of the other project views will be an extension of one of the two views previously described. This is done by using the expression @extends(view_name) which allows to inherit all the code defined in that view. It replaces the fields identified as @yield with the existing fields identified with @section.

The following blocks of code show an example of a view of the public section of the website (first code block) that inherits a main view (second code block).

Listing 3.1: Example of a main view that will be inherited by other view.

```
<!-- about.blade.php Code -->

@extends('main')

@section('content')
 <!-- Code -->

@endsection
```

Listing 3.2: Example of a view that will inherit a main view.

```
<!-- main.blade.php Code -->

<!DOCTYPE html>
<html>
<head>

 <!-- Code -->


</head>

<body id="home">


 <!-- Code -->


        <div class="ui vertical feature segment">
                <div class="ui centered page grid">
                        @yield('content')
                </div>
        </div>


 <!-- Code -->


</body>
</html/
```

Figure 3.8 shows an example of a page (homepage) created with the method described before. The section [a] of the figure is general content that can be used in several views so it is created in the main view. The homepage view has its own content, shown in the figure with section [b], but also extends the main view, resulting in the header and footer created in the main view, section [b].

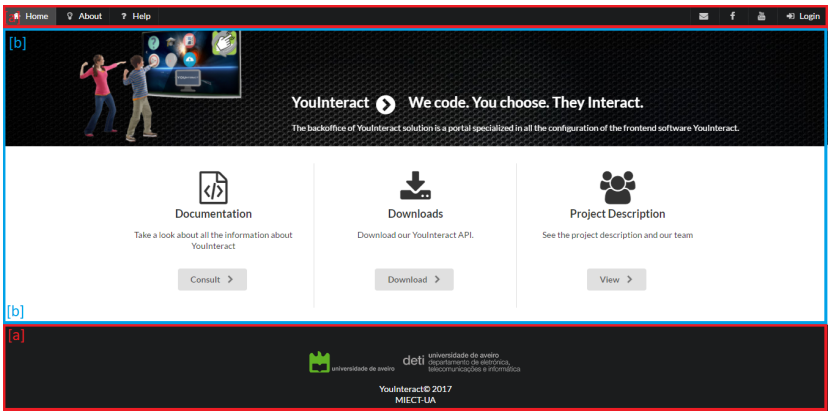


Figure 3.8: Example of a view that inherits a main view where section [a] is created in the main view and section [b] created in that view.

3.4.4 Controllers

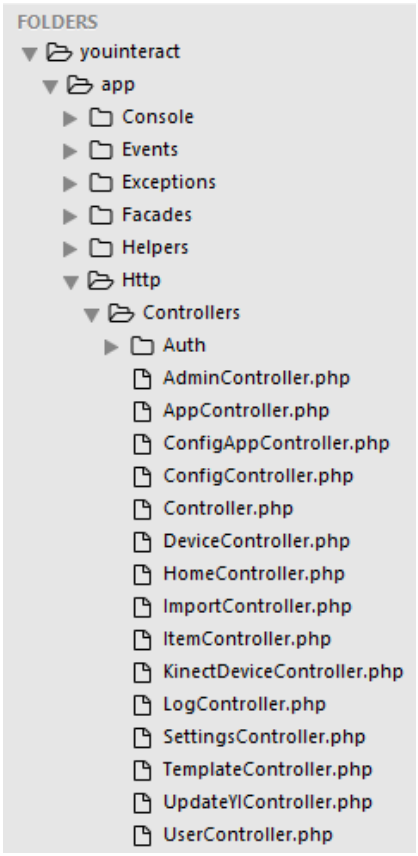


Figure 3.9: Controllers organization of the Portal.

Figure 3.9 shows all controllers present in the application. Each one of them may have several purposes, mainly regarding interacting with the database by receiving or modifying information. Every Controller file is available at app/Http/Controllers and there is at least one Controller for each existing Model.

Usually, each Controller has at least these five methods.

- index- Forwards to the view index which normally shows the user specific elements of the database.
- editIndex- Forwards to the vew editIndex where the user is able to modify some of the data.
- add- After the user submits the new data, these are validated and then saved in the database.
- edit- After the user modifies the data, the changes are validated and the updated in the database.
- delete- Used to destroy specific data chosen by the user.

3.4.5 Database

YouInteract Portal database is built with mainly nine tables (Figure 3.10). The ”users” table has the information about all of the users registered in the website and the ”logs” table has the information about the changes each user has made in different parts of the website.

”Devices” table has different kinds of data belonging to the devices that are running YouInteract and each device has its own configuration (”configs” table). Each configuration has its own template (”templates” table) with information regarding the background image. It may also contain information about items that are only available at specific configurations (”config_items” and ”items” tables). Finally, each configuration has the information about what applications are available and, if needed, their configuration (”config_apps” and ”apps” tables).

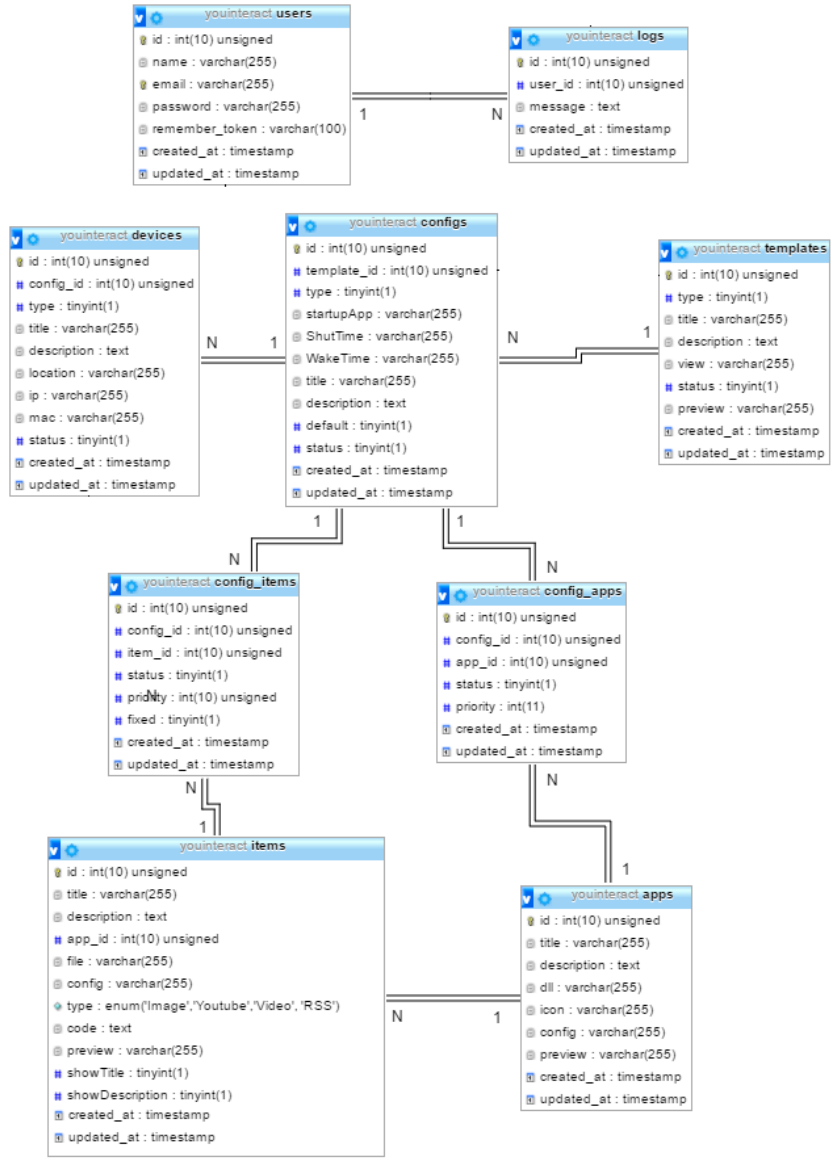


Figure 3.10: Portal’s database diagram.

The interaction with the database is done through Eloquent ORM [Laravel-eloquent], a tool provided by Laravel framework where each table present in the database has a corresponding Model. This way, every request to the database is not done using SQL Queries [Laravel-queries] but instead using methods that are much easier to implement. The following example shows how it is possible to return a specific device present in the Device table by passing the desirable ID.

Listing 3.3: Example showing how to get a device with a specific id from the database.

```
public function example1()
{
        return Device::find(id);
}
```

It is also possible to combine several methods in order to create a more complex request. The following example shows how to retrieve, from different tables of the database, the existing devices, the configurations title of each one and their templates title and id.

Listing 3.4: Example showing how to get an array of devices configurations and their templates from the database.

```
public function example2()
{
  return DB::table('devices')
  ->leftjoin('configs', 'devices.config_id', '=', 'configs.id')
  ->leftjoin('templates','configs.template_id','=', 'templates.id')
  ->select('devices.*', 'configs.title', 'templates.id',
  'templates.title')
  ->get();
}
```

The database maintenance is done through Migrations [Laravel-migrations], a tool that helps to manage the tables present in the database. It includes two methods: one for executing the migration (up) and the other for when the migration is canceled (down). The following example shows that when the migration is executed it will create a table named ”apps” with seven different parameters and when the migration is annulled, the table is eliminated from the database.

Listing 3.5: Example of a migration of a table to the database.

```
<?php

use Illuminate\Database\Schema\Blueprint;
use Illuminate\Database\Migrations\Migration;

class CreateAppsTable extends Migration
{

        public function up()
        {
                Schema::create('apps', function(Blueprint $table)
                {
                        $table->increments('id');
                        $table->string('title');
                        $table->text('description');
                        $table->string('dll');
                        $table->string('icon');
                        $table->string('config');
                        $table->string('preview');
                });
        }

        public function down()
        {
                Schema::dropIfExists('apps');
        }

}
}
```

There are four different methods that could be evoked in the command line related to Migrations.

- php artisan migrate: Executes all of system's Migrations.
- php artisan migrate:rollback: Undoes all changes until the last operation.
- php artisan migrate:reset: Undoes all changes.
- php artisan migrate:refresh: Undoes all changes and executes all of them again.

In order to insert a large quantity of data in the database for a purpose of testing or for fill the website with initial values important for a first utilization of it, a Seeder tool can be used [Laravel-seeders]. To do it, the DatabaseSeeder.php file must be changed with the desired data and then use in the command line the command "php artisan db:seed". The following example shows a way that uses Seeders to insert four applications in the Applications table.

Listing 3.6: Example of how to seed a table with DatabaseSeeder.

```
<?php
use Illuminate\Database\Seeder;
use YouInteract\Models\App;

class DatabaseSeeder extends Seeder
{

  public function run()
  {
        App::create(['title' => 'You_Tutorial',
        'description' => 'You_Tutorial descricao.',
        'dll' => 'You_Tutorial.dll',
        'icon' => 'You_Tutorial.png',
        'config' => 'You_Tutorial.xml',
        'preview' => 'You_Tutorial.png']);

        App::create(['title' => 'You_AirPaint',
        'description' => 'You_AirPaint descricao.',
        'dll' => 'You_AirPaint.dll',
        'icon' => 'You_AirPaint.jpg',
        'config' => 'You_AirPaint.xml',
        'preview' => 'You_AirPaint.png']);

        App::create(['title' => 'You_Videos',
        'description' => 'You_Videos descricao.',
        'dll' => 'You_Videos.dll',
        'icon' => 'You_Videos.png',
        'config' => 'You_Videos.xml',
        'preview' => 'You_Videos.png']);

        App::create(['title' => 'You_Gallery',
        'description' => 'You_Gallery descricao.',
        'dll' => 'You_Gallery.dll',
        'icon' => 'You_Gallery.jpg',
        'config' => 'You_Gallery.xml',
        'preview' => 'You_Gallery.png']);
  }

}
```

3.4.6 YouInteract Communication

All of the described components of the Portal were built in order to guarantee the main goal of having a configurable YouInteract application through a website Portal. After explaining how an administrator can configure and adapt the YouInteract according to his needs, it is important to understand how the communication between the Portal and the YouInteract application is made.

As explained before, YouInteract files are present in a Dropbox account which the system administrator has to configure in the computer that will run YouInteract (Appendix F). This method is used to ensure that every device running YouInteract is up-to-date and the Portal and YouInteract are always synchronized. Every file that must be available in the computer's file system directory where YouInteract is running, is saved there using the Flysystem PHP package available in Laravel framework. When an administrator adds new items, applications and themes, their data information is validated and saved in the database and then the files are uploaded in the right directory of the cloud account file system.

In order to use Flysystem PHP package, it is necessary to indicate to the Laravel framework the dropbox's name account and its security token in order to have access to that same account. This is done in the config/filesystem.php file. After having this configured, it is now possible to upload files to that dropbox account using the Flysystem PHP package as the following example shows.

Listing 3.7: Example of uploading a video to the dropbox account.

```
\$cloud->put("/videos/".\$uniqid."."Request::file('attachmentVideo')
->getClientOriginalExtension(),
file_get_contents(Request::file('attachmentVideo'))));
```

In addition to these files, the device that runs YouInteract must also have the configuration file that will personalize its behaviour. Since each device may have its own configuration, these type of files can not be in the Dropbox account to allow different configurations in different computers. To overpass this problem, it was decided that every time the YouInteract application ran, it would contact the Portal, receive the configuration files and save them in a personal directory of the computer's file system: C:/Users/[NameOfUser]/AppData/Roaming/YouInteract. Through this method, in addition of guaranteeing that every computer running YouInteract has its own configuration files, it is also guaranteed the system will work without internet connection.

As Figure 3.11 demonstrates, every time the website receives a POST with a MAC address to the address: "/device/kinect/get", it will check in the database if there is a device with that MAC. If there is not, it will add it and then send the configuration XML file in a response to the POST.

In addition to the general configuration file, it is also possible to have configuration files for a specific application. The method used is the same, so when the Portal receives a post to the address: "/device/kinect/app/get" with the MAC address and the application name, it will send back the configuration XML file in a response to that POST.

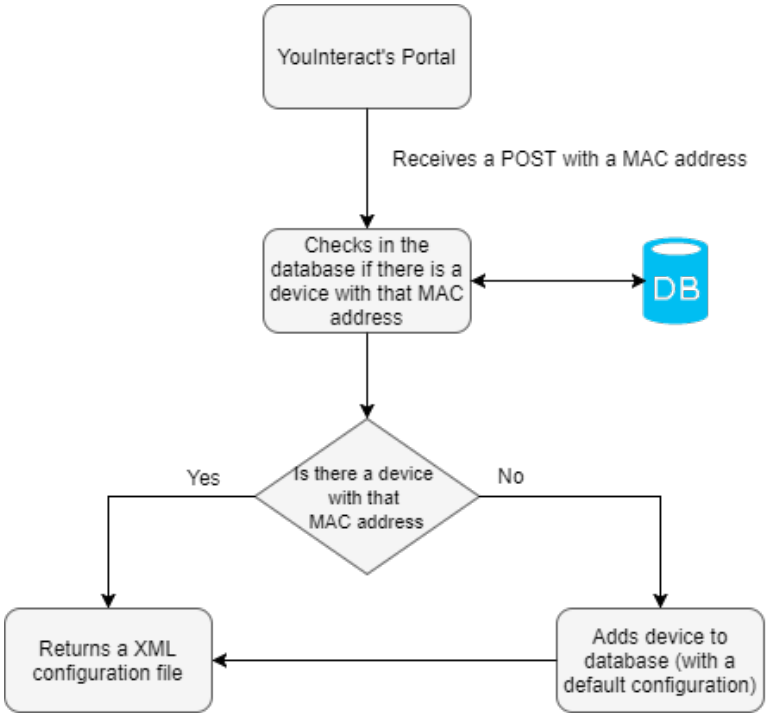


Figure 3.11: Diagram showing what happens when a computer is communication with the Portal.

Chapter 4

YouInteract:Core

YouInteract aims to be a core software application, possible to be configured through a portal (see Chapter 3), that support gesture interaction while integrating several applications depending on the context of use.

4.1 Concept

This project started in 2009 with a system capable of displaying relevant information to DETI's students and followed this path until this dissertation. Since the first versions, DETI-Wall and DETI-Interact, it was decided it could be used in several other contexts, reaching other target audiences.

With this goal in mind, the project's name was changed to YouInteract giving the idea that anyone could have a reason to interact with the system.

With a computer, the YouInteract application installed in it, a display and a Microsoft Kinect V1 (Figure 4.1), the system administrator is able to run a personalized application (see Appendix F on how to set up a YouInteract device). This customization is guaranteed by choosing the desired configuration and applications in a specific web portal (see Chapter 3). In this way, the administrator can configure each group of PC's that have YouInteract installed in a way that it adapts its contents to the locations.

Figure 4.1: Hardware needed consists on a display, a Microsoft Kinect and a computer with Windows as an operative system.

Based on this concept, a system architecture was designed to respond to the project's needs. Figure 4.2 presents the most important components integrated and implemented in the final solution.

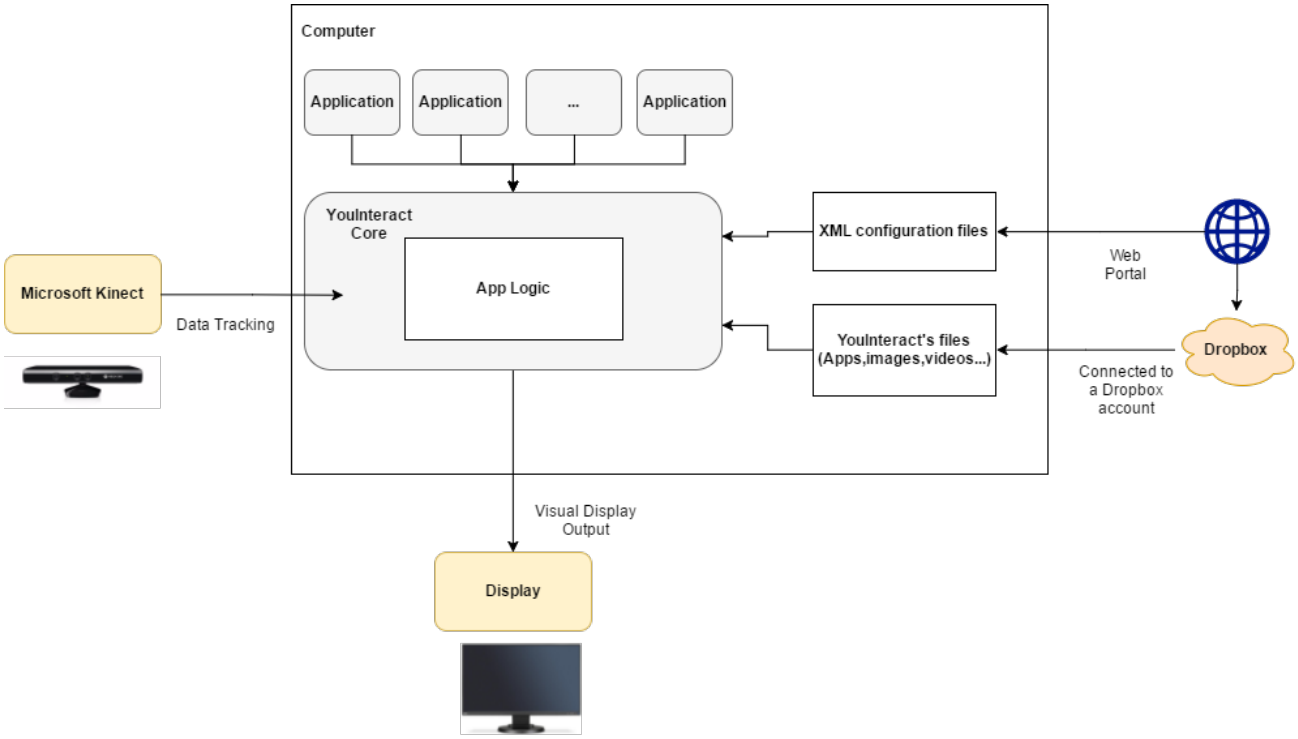


Figure 4.2: System architecture of the main application.

The system architecture detailed in Figure 4.2 gives an overview of the hardware and software components connection. With an input device and an output device, Microsoft Kinect and a screen display respectively, the system works with all of its features.

The input device and the connection to the Web Portal work independently since if no input device is connect, the system will also work as an information display, without gesture interaction. In addition to this, if no network is available, the system will load the configuration received in the previous internet access (that was stored locally), or will use a default configuration if none internet access happened. It was designed this way in order to guarantee a fully working system every time the application is launched even though it might not have an Internet connection.

As Figure 4.2 shows, the major YouInteract components are:

- Communication with the Portal deals with the exchange of information between the device running the application and the Portal.
- Integration of Microsoft Kinect is the block responsible for accessing the data from Kinect sensors.
- The application's core.
  - Application's interface: how the information is displayed to the user.

  - Applications' integration: how the main application can integrate other applications.
  - System navigation: manager navigation between applications is made.
  - Main behaviour: how the system conducts while its running.
- The different applications that can be integrated.

The next sections of this chapter will describe the technologies used to implement this architecture and its components as well as a detailed explanation on how and why each of the previous described components were created.

4.2 Used Technologies

The first stage of this project explored and analyzed different software solutions that would serve as a base for the desired system. It was fundamental that all of the chosen technologies had good documentation, were not only easy to use but also to learn and finally, that they had a good community support.

With this ideas in mind, the following technologies were selected to create the YouInteract system.

4.2.1 Microsoft Kinect SDK 1.8

There are two available options to integrate Microsoft Kinect in a Windows environment: Microsoft Kinect SDK and OpenNi. The first option is the simpler and most used approach since it was designed by the hardware creators. Several official sample projects are available as well as good quality documentation and it is now in version 1.8, for Microsoft Kinect v1, with several years of development from Microsoft. OpenNi has the big advantage of also working in a Linux environment, which Microsoft's SDK can not. It would also be a good option, however the documentation is not as good as Microsoft's and does not offer as many interaction options as its rival.

With all of this in mind amd since YouInteract was not designed to work in Linux environments, it was decided to use Microsoft Kinect SDK v1.8.

Why not use a Microsoft Kinect v2 and its SDK 2.0? Firstly it is important to remember the project's requirements that said YouInteract needed to be an application able to run on low cost computers. After a comparison between the system requirements of Kinect v1 [v1 requirements] and Kinect v2 [v2 requirements] it is possible to understand that Kinect's v2 requirements do not match this project's requirements since it demands expensive hardware. In addition to this, if Kinect v2 was chosen, it would be needed to create a package in Unity in order to integrate this version of Kinect with the game engine since the asset available in the Unity store is no free, which would take too much time. Despite all this, and since it was possible to have a Kinect v2 and explore its SDK for a week, an approximated application core for this Kinect's version was created. It can serve as an initial step to future work since, for now, it can only show a menu with the existing five WPF applications (see Chapter 6). This limitations happened because of all the differences between SDK 1.8 and SDK 2.0 (differences in the Kinect region, skeletons streams, buttons responsive to the virtual hand, among other). It was impossible to have, in useful time, an exact copy of YouInteract for Kinect v2.

4.2.2 Windows Presentation Foundation

One of the main requirements of the project was that this application was able to run in most of PCs available on the market. Since Microsoft Windows is the most used operative system [OS-Marketshare] in the world and it offers an easy way of integrating Kinect SDK, it was obvious that YouInteract needed to run on this environment. Windows Presentation Foundation (WPF) is the principal graphical environment tool designed by Microsoft for developers being able to create a Windows application.

WPF is a component of Microsoft .NET Framework and its user interface is created using Extensible Application Markup Language (XAML) and its behaviour is coded with C#.

4.2.3 Unity support

One of the main goals for this new version of YouInteract was that the system was able to integrate applications created with game engines. This way, the system could offer the user a chance to interact with 3D applications which would be extremely relevant for creating educational and playful applications for FCCVA. In addition to this, it would also allow to integrate previous 3D applications developed in other department's dissertations, such as J. Cardoso dissertation [Cardoso, 2015];

Since there are two major cross-platforms game engines, Unreal Engine and Unity, a decision in which type of applications YouInteract would be able to support needed to be made. Although both engines have high quality documentation and an active community, Unity has more available tutorials and sample projects that help to decrease the learning curve of the development process.

It also has support for Microsoft Kinect via an unofficial "unitypackage" developed by several members of the Unity community, available in the Unity store for free.

Finally, Unity provides native and more robust methods to integrate an application made with this game engine in a WPF application.

4.3 Communication with Portal

The previous version of the system already integrated a Portal used for basic configuration so it was decided that we should take advantage of the existing good code. After analyzing and testing, it was concluded the communication with the Portal of the previous version was efficient and scalable, so in this version, although it has changes and additions, such as adding a default configuration to ensure the system works all the time, it was decided to continue the good work and its structure is the same.

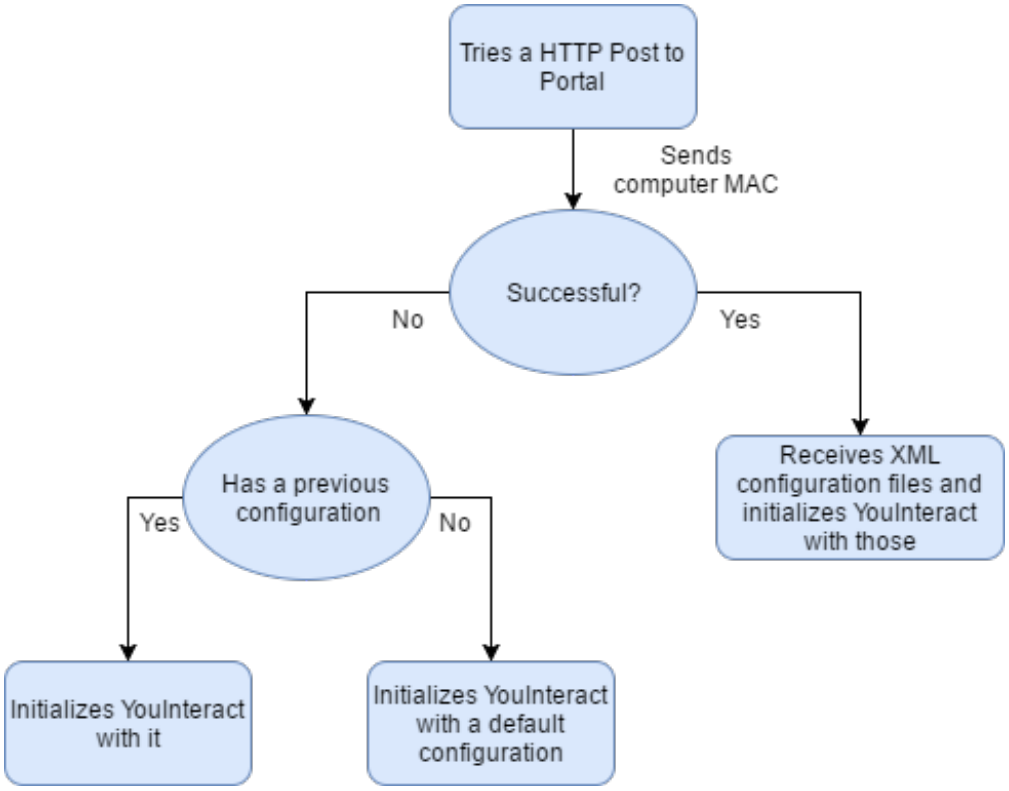


Figure 4.3: Flow diagram of the system communication with the Portal.

As described in Figure 4.3, the system begins to check if it can connect to the URL associated with YouInteract Portal in order to get a XML configuration file. It performs a HTTP POST to a specific URL where it sends the PC's mac address. If this connection is not successful, it will load the last configuration retrieved that is stored locally, or if it has none, it will load a default configuration saved in the YouInteract's files.

If it is successful, it will save the specific configuration files in a personal files directory, used later if the internet connection is lost: "Users/UserName/AppData/Roaming/YouInteract/XMLAccess"

By using this strategy, it is guaranteed that even without Internet connection, YouInteract will fully work.

There are two types of configuration files. The first one is a XML file containing information regarding general system options. Among several other information, it indicates which applications were chosen to be available, which is the background image for the main Menu and which is the startup application.

The structure of this XML file is described next.

Listing 4.1: Structure of the main configuration XML file.

```
<KinectApp>
        <Themes>
                <Theme_id>"Theme's ID"</Theme_id>
                <Theme_name>"Theme's name"</Theme_name>
                <Theme_file>"Theme's file path"</Theme_file>
        </Themes>
        <Apps>
@foreach chosen app
        <Entry>
                <Apps_name>"App's name"</Apps_name>
                <App_id>"Apps ID"</App_id>
                <App_icon>"Apps Icon file path"</App_icon>
                <App_desc>"Apps description"</App_desc>
                <Dll>"Name of App's Dll"</Dll>
        </Entry>
@endforeach
        </Apps>
        <StartupApp>
                <App_name>"Initial app's name""</App_name>
        </StartupApp>
        <AutoShutdown>
                <ShutTime>"" if none or "Hour:Minute"</ShutTime>
                <WakeTime>"" if none or "Hour:Minute"</WakeTime>
        </AutoShutdown>
</KinectApp>
```

The second type of configuration file is related to single applications. Most of its structure is defined by the developer of that application however it must respect two rules.

- The file's name must always be the name of the application (e.g. You_Images.xml).
- The first node of the XML must be "config" to indicate that a file is a configuration file for that specific application.

The following example show a configuration file of an application that display images in a slide show manner. The computer could have multiple images stored in the specific directory that YouInteract is reading however with this configuration file, the system will only load and show these two images as described next.

Listing 4.2: Structure of an example of an application's configuration XML file.

```
<?xml version="1.0" encoding="utf-8"?>
<config>
  <image>
        <title>Photo 1</title>
        <path>foto2.jpg</path>
```

```
        <description>Example 1 photo</description>
    </image>
    <image>
        <title>Photo 2</title>
        <path>photo2.jpg</path>
        <description>Example 2 photo</description>
    </image>
```

The communication only occurs once, when YouInteract is initiated in order to prevent inconsistencies that could appear when a configuration is modified in the portal while the system is running and being used.

4.4 Gesture interaction

The second step in YouInteract initiation is preparing and starting everything related to the hardware responsible for gesture interaction, the Microsoft Kinect. Microsoft Kinect SDK was fundamental since it is the easiest and most complete way to get all the available functions present in this equipment.

Firstly, the Kinect sensor is started and the four Streams that Kinect can provide are configured.

- Depth Stream: depth data and player segmentation data in each frame.
- Skeleton Stream: data of the detected skeleton's track in each frame.
- Color Stream: color image data.
- Interaction Stream: processed data from the Skeleton Stream which gives data about, among other things, the hands of the user.

Since the documentation of Microsoft Kinect SDK about performance and memory management was almost non existing, the management of the four types of streams described before was in constant change in order to pursue the best way of dealing with memory and CPU usage.This search for a solution that would be efficient and at the same time saving the most possible computer resources could be divided in two phases.

Initially, all these streams were started and kept running until the application's close even if they were not used. This led to drastic performance problems (CPU and memory usage). Since one of the project's requirements was to ensure YouInteract was capable of running not only on modern computers with expensive hardware components but also on average computers that fulfill the minimum system requirements to run a Microsoft Kinect [2], a better approach to the problem was needed.

A new approach that minimized those performance problems was designed to use only the necessary streams at a given time. A rule was made that every application developed must indicate, through a function present in the YouInteract API (Section 5.2), what streams it needs to properly function. With this rule, and since the main menu of YouInteract only needs the Depth and Skeleton data information, the rest of Kinect's Streams are disabled until a specific application requires it.

4.5 WPF Applications integration

An important goal of the project was that anyone with simple knowledge in coding with C#, and specifically in WPF, would be able to develop new Apps for the YouInteract system. Two different ways of integrating a WPF application in our system were investigated.

- Having the App in an executable file and importing it on WPF.
- Having the App in a DLL file and integrate it in YouInteract project.

The DLL file approach was selected because the developer did not need to know how to deal with Kinect and its SDK since it is already initiated in the YouInteract roots. We also have more control of the application in this way since the .dll file is loaded when YouInteract starts so if (with a method called [Reflection], for some reason, an error is detected, the application is disabled and the system consistency is maintained. Finally, with this method, an YouInteract API was developed to make the job of the developer easier (Chapter 5).

With this strategy, all the applications DLLs files are in the same directory as the YouInteract root files but only the active applications (the ones present in the XML configuration described before) are loaded. The system begins to compare the active applications from the configuration file with the DLLs files available (must have the same name) and if it exists a match then the name of the application will be added to the system navigation system (explained in the next section) and the application will be ready to be used. If there is no match, it means that whether a specific App is not active or the DLL file is not available. If this occurs the system will ignore the App and will move to the next. Finally, each of the integrated Apps will have an icon shown in the Main Menu ready to be pressed by the user to open and interact with it.

4.6 Application Interface

[Hinrichs et al., 2013] state that the diversity of public displays and the context in which they are inserted present different requirements regarding interface design. However, YouInteract must be able to adapt to different context so it needs to have an interface that can be understood in most places.

As stated before, there was an analysis on good implementations from previous versions of YouInteract so, in addition to having continued the already developed structure in the "Communication with the Portal", it was also decided to maintain the User Interface from the previous version of the system. It was an interface developed with the help of designer students with good feedback from the students that said it was clean and easy to understand. With this in mind, no big changes were made with the exception of now being able to customize certain icons and images of the system through an offline mode.

The system begins by showing the main Menu which is a special page in the structure since it is the core of the interaction with the user. In this page, all the icons of applications available are shown to the user and he can choose which one he wants to run by using one of his hands to press the wanted icon. The connection between the user's virtual hands representation and the applications icons is done with the help of Kinect Region of the Kinect SDK.

The Main Menu (Figure 4.4) exhibits a maximum of eight icons at the same time and if there are more than eight available applications, two arrows are shown at the edges of the

screen to indicate the user that it is possible to go to a different page of the Menu containing more icons. This change of page is possible by closing the hand and slide it to the desired side. It is a system inspired on the smartphone's operative systems (e.g. IOS and Android) that use the "Swipe gestures" to navigate between menus.

The background image and the App's icons can be fully personalized since the Web Portal (described in the Chapter 3) offers the possibility to the system administrator to upload and choose the intended images.

Regarding the App's Interface, the developer can select to implement it and design it the way he pretends.

Figure 4.4: YouInteract's main menu interface.

4.7 Application Navigation

YouInteract's system was inspired in the mobile most known operative system, such as Android and IOs, presenting a menu with several different applications that can be launched. So, in YouInteract, the first page is the Main Menu that offers the possibility for the user to select which application wants to open. When an App is chosen, it will then navigate to the initial page of that App and according to the decisions of the user it will continue navigating through the App's pages or go back to the Main Menu.

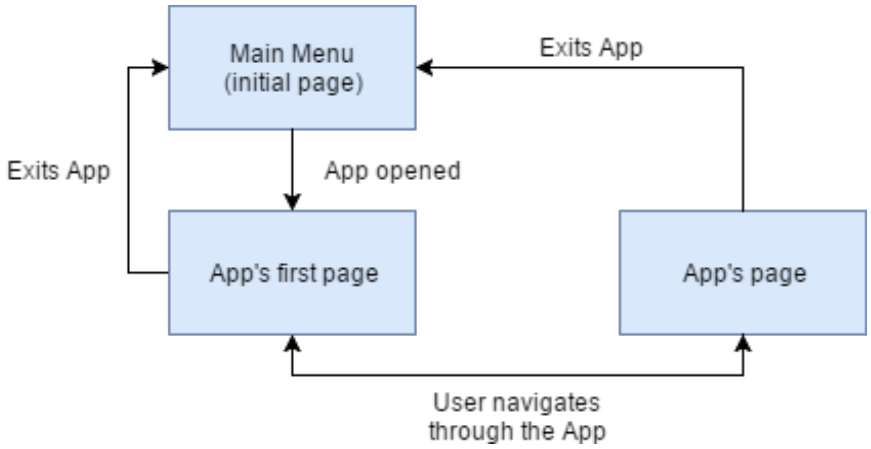


Figure 4.5: Navigation inside YouInteract.

As it is possible to understand from the Figure 4.5, a navigation system between the Apps and a different navigation system inside the App, namely between the pages of that App, was necessary.

WPF has a mechanism named ”Frame” that allows the developer to navigate between different pages of the application. Since all applications are integrated as soon as YouInteract is started, the system has all the information regarding the pages used by each of the applications. So, when a user selects an App, the system checks what its first page is and its requirements. With that information, it updates the sensor requirements (as described before) if needed, binds the Kinect Region to the Kinect Sensor and unbinds it from the main Menu and finally tells the WPF Frame to navigate to the first page of the chosen application.

When the navigation is inside the application, the method used is the same. The system checks the requirements for the new page, unbinds the sensor from the previous page and binds it to the new (if needed) and finally the WPF frame is set to navigate to the page with a specific name.

4.8 YouInteract Main behaviour

YouInteract system should be flexible enough to respond in different ways according to different inputs received therefore the system works as a state machine with several states during its running-life and changes its state according to the type of input.

The strategy was to implement a solution, test it by observing users interact with it, discover details to be improved, implement them and test the changes again. This method was used approximately over three weeks and can be divided in following three phases.

- First phase: Menu and Screensaver: when a user was recognized, the system would immediately changed the Screensaver view for the main Menu view.
- Second phase: Menu and Screensaver + Lift hand: when a user lifts the arm, the system shows the main Menu.
- Third phase: Menu and Specific App + Lift hand + Press Button: the system would begin by showing a specific app. When the user was recognized, he needed to lift the arm and then press a button to go to the main Menu.

In the first phase, the YouInteract system would start by displaying the Main Menu and if no user is detected for a certain amount of time (10 seconds by default), it would launch a Screensaver which was only a slideshow of images. When it was displaying the slideshow, if a person walked in front of the setup, the system would display immediately the Menu again (Figure 4.6). Right on the first test it was concluded this process is not adequate since there are several people passing in front the setup that do not have the objective of interacting with it so it was needed to find a way to overcome this situation.

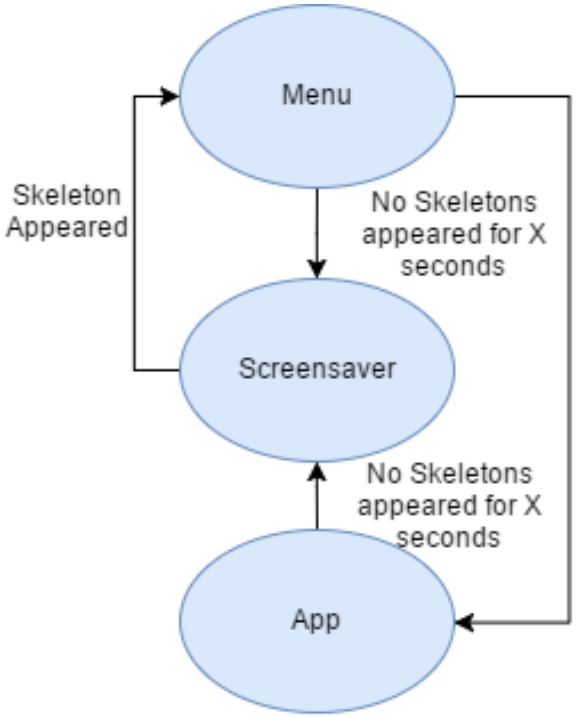


Figure 4.6: First version of the behaviour implemented on YouInteract where the Menu would show up every time a person went by the system.

The first strategy to overcome people passing in front of the setup that do not intend to interact with it was to find a method to evaluate users' intent to interact with the system. So, in the second phase of the behaviour system, a strategy was developed to force the user to lift the arm in order to access the main Menu (Figure 4.7). This way, the Screensaver would disappear and Menu would be shown. This method helped decrease a lot of changes to Main Menu when no one wanted to use the system. However, it continued to have several false positives since the setup was installed in a lobby where people used to wait so detecting lifted hands was usual even when a person was not using YouInteract. It was again concluded the system needed to be even more robust.

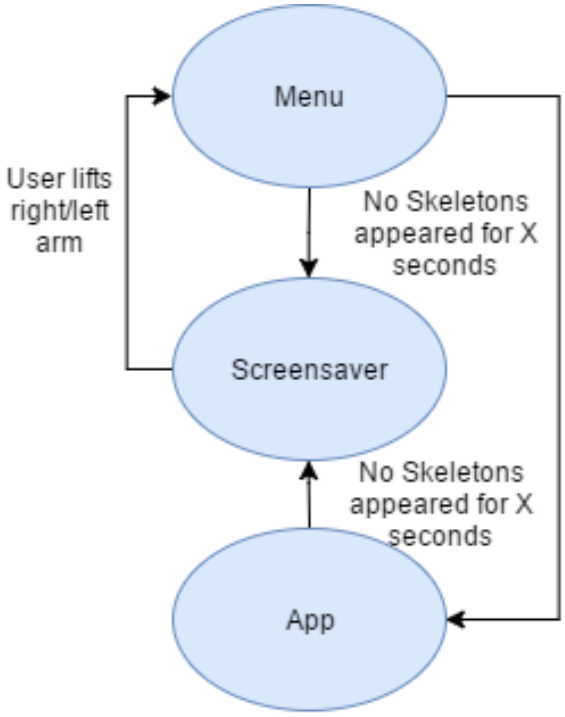


Figure 4.7: Second version of the behaviour implemented on YouInteract where the Menu would only show up if the person lifted the arm.

In third phase a button was added for the user to press if he desired to interact with the system (Figure 4.8). When the system is displaying the Screensaver, the user must lift his hand where it will show a button that the user needs to press in order to go to the Menu. This is a more advanced and complicated method but it ensures the system will only show Main Menu when there is a user pretending to interact with it.

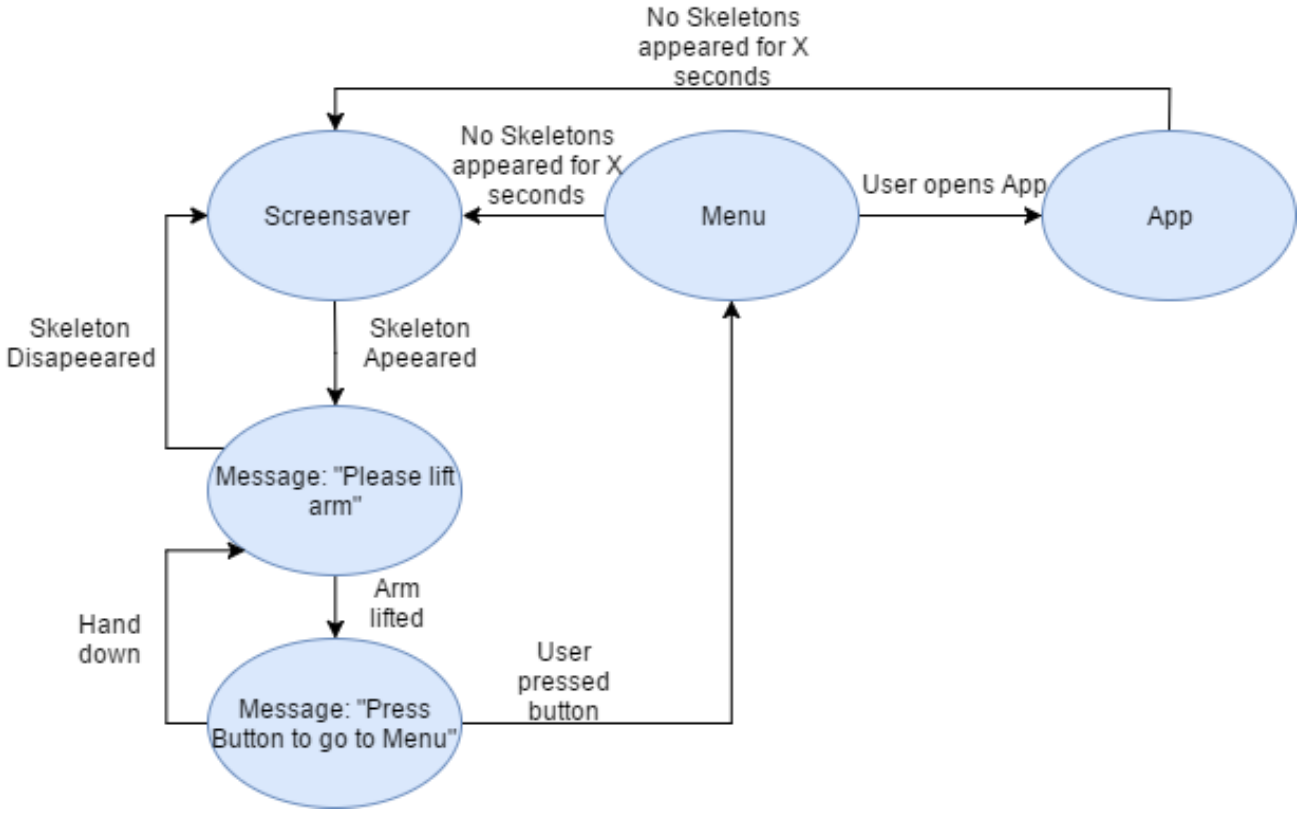


Figure 4.8: Second version of the behaviour implemented on YouInteract where the Menu would only show up if the person lifted the arm.

Finally, YouInteract had all features which made possible to have a robust system. However one more change was made related to the Screensaver. Instead of mandatorily presenting Screensaver when no one is using YouInteract, it was given the possibility to the system administrator to choose which application was launched when no one was interacting. If no application is selected then YouInteract functions as in phase one with just Menu and Screensaver. On the other hand, if the system administrator selects in the Web Portal a specific startup application the system will work as described in phase three where the system will launch the chosen App when nobody is using it and to return to the Menu, the user needs to lift his hand a press the shown button as shown in Figure 4.9.

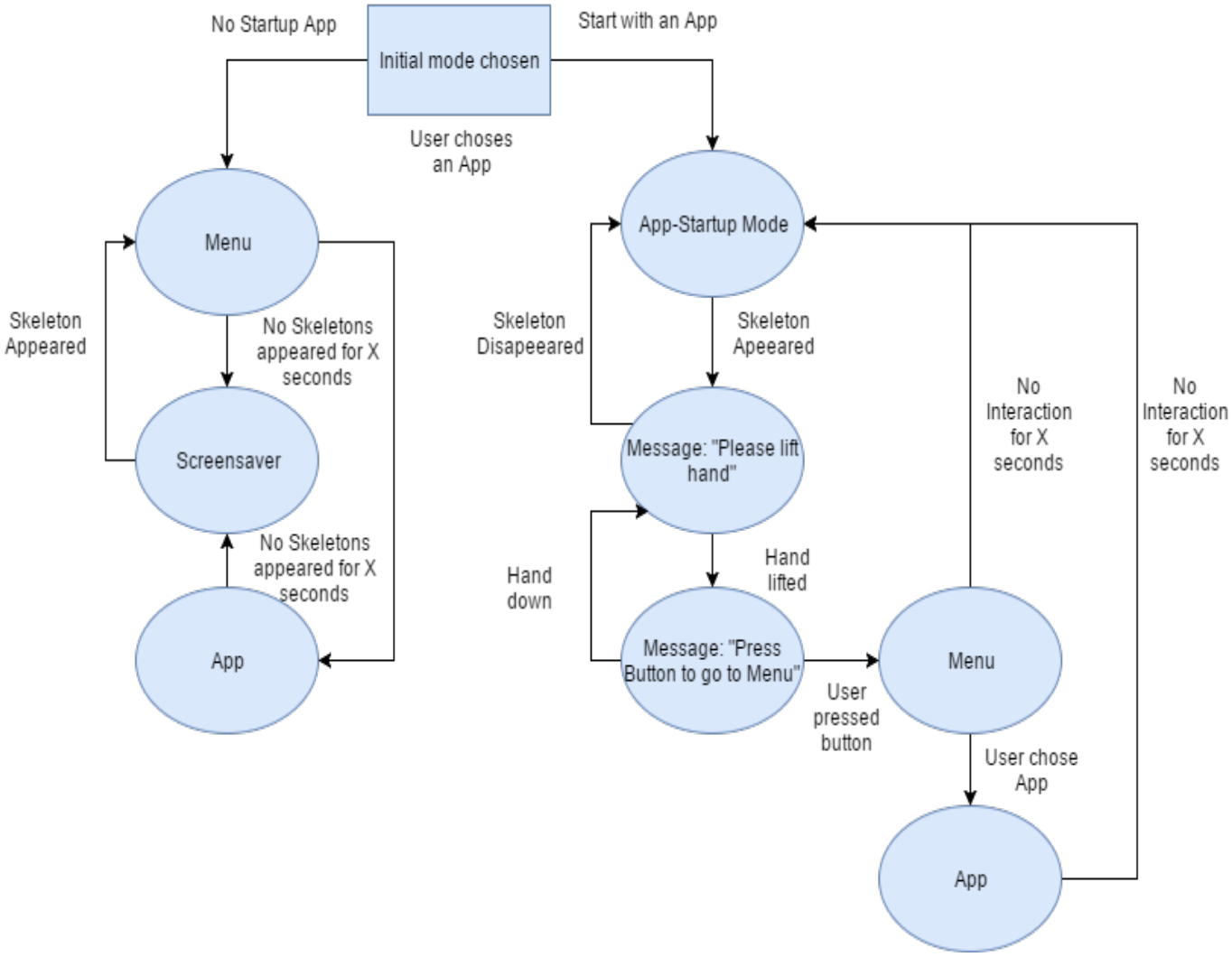


Figure 4.9: Final version of the behaviour implemented on YouInteract.

4.9 LOG

For debug and monitoring, an objective of the work was to add logging capabilities to YouInteract. With this in mind, YouInteract's log was created.

YouInteract's log consists on a .json file with records of all events regarding the main system for each day YouInteract ran. In addition to this, a YouInteract's application developer may also use methods available in YouInteract API to create a log file regarding this specific application. To sum up, YouInteract mandatorily creates a general data file for each day and also may have data files for each active application if the developer implemented that function.

So, every time YouInteract is launched, a folder with the current date is created and inside it, folders for each active application and the main log file are created (Figure 4.10).

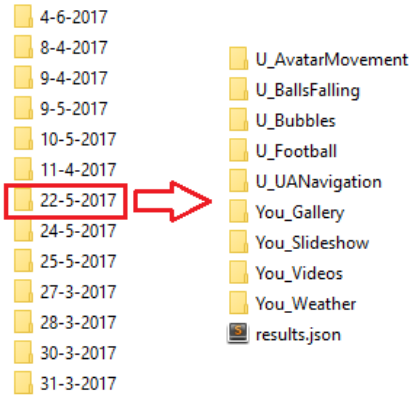


Figure 4.10: Organization of the directories of the Log.

The main log file shows important data to help system administrators to know if the system is running without any problems, the most used applications and the number of users interacting with the system.

Log file information:

- Time the application was launched.
- Number of people captured.
- Number of people who interacted with the system.
- Number of times each YouInteract application was initiated.
- Time an application was initiated and time it was terminated.
- Number of alerts due to high percentage usage of the computer's processors and its time.
- Time the application was shut down.

With all of these parameters, a system administrator can know directly or indirectly if YouInteract crashed any time (if the log does not show when the system was shut down, it means it crashed), the most used application, concerns about the processors' usage resources,

among others. This log file structure is organized and easy to read since its implementation was thought to enable the person responsible to analyze the data, to being able to read directly the data from the file without any third-party software (like Microsoft Office Excel).

The following shows how the file is organized.

Listing 4.3: Structure of the main log file.

```
{
  "Running": {
    "WakeHour": "Time the application was launched"
    "N_Skeletons": "Number of people captured"
    "Skeletons_Hour": [ "Time X person was captured",
    "Time X person was captured",
    "Time X person was captured", ...
    ],
    "N_Interactions": "Number of people interacted with the system"
    "Interactions_Hour": ["Time X person started to interact",
    "Time X person started to interact",
    "Time X person started to interact", ...
    "AppsUsed": {
      "App's Name": {
        "NTimes": "Number of times the App was launched",
        "Time": [ "Time of launch - Time of exit",
        "Time of launch - Time of exit", ...
        ]
      },
      "App's Name": {
        "NTimes": "Number of times the App was launched",
        "Time": [ "Time of launch - Time of exit",
        "Time of launch - Time of exit", ...
        ]
      },
      ...
    },
    "N_CPUAlerts": "Number of times it happened a CPU alert",
    "CPUAlerts_Hour": [ "Time of CPU alert",
    "Time of CPU alert", ...
    ],
    "ShutHour": "Time the application was shut down"
  }
}
```

4.10 Other Features added

After initial testing (see Chapter 6), there were some interaction problems that lead to the creation of new features. Also, some of the following features were suggested by employees of FCCVA as well as from people working at the university department where YouInteract was installed, DETI (Departamento de Electrónica, Telecomunicações e Informática). These features did not involve changes in the system main behaviour however they added potentialities to YouInteract.

4.10.1 Timer Click and Push Click

Over the two days of tests, it was possible to observe a considerable amount of users with difficulties to select specific buttons in YouInteract (see Chapter 6 for details). This was due to the fact that with Microsoft Kinect 1.8, in order to select a button, the user needs to make a hand press gesture (Figure 4.11). Although this gesture is easy to be done, it involves a learning curve for inexperienced people (that might extend the arm and thus cannot push the button any further).

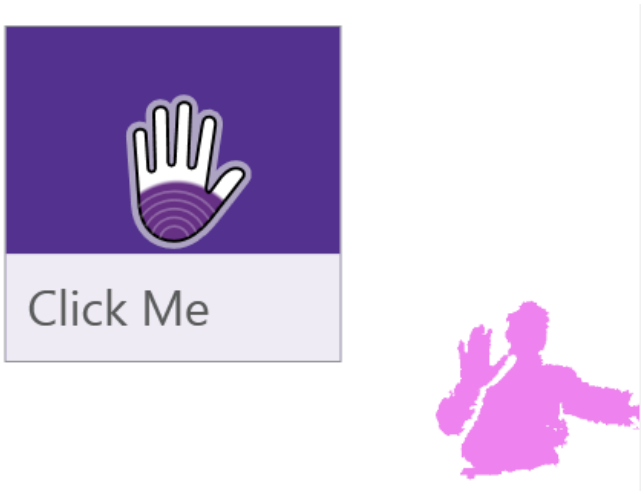


Figure 4.11: Gesture a person has to do in order to press a button.

In order to minimize the problem and after analyzing different related projects (Chapter 2), it was decided to implement a Timer click. When a user overs a button with the virtual hand, a progress bar is displayed, indicating the user that the longer the time the user leaves the hand over the button, the more filled it gets (Figure 4.12). When the progress bar is full, a button click is done and that button is selected. The time needed for a click and the color of the progress bar may be configured in the YouInteract Portal while its size can be defined by the application's developer using YouInteract API (see Appendix C for details).

The two methods coexist and the user may opt to push the hand for a click or just stand over a button until the progress bar is full.

Figure 4.12: Two methods to select a button: Timer Click and Push Click.

4.10.2 Video Tutorial

Over the two days of tests, it was possible to observe that some users did not know how to start interacting with the system and needed a brief explanation on how to do it. To minimize this problem, a short video tutorial was created and is displayed when the detected user is not making any action for a given time (Figure 4.13). As soon as the user successfully performs an interaction, by for example opening an application, the video immediately disappears.

Figure 4.13: A short tutorial video showing how to interact with the system is displayed if a user does not perform any action for a given time.

4.10.3 Shutdown and Hibernate/Awake

In previous versions, YouInteract needed to be launched every day manually by someone. It was concluded that if possible, an automatic mechanism that could hibernate and then awake the computer would offer good value to the product.

An administrator can now choose in the Portal the time to hibernate and then awake or, if the administrator prefer, the time the computer shuts down completely. Although hibernate and shut down were possible to be done without problems, awaking the computer after being hibernated was more complex. To overcome this problem, a Windows file named kernel32.dll was used. It offers the possibility to invoke two functions that create an entry in the Windows timetables with the date and time to awake the computer. However, it is important to register that it must be given permission to the Windows to awake the computer. That can be done by executing the following steps:

- Go to "Control Panel
- Select Power Options
- Click on "Change Plan Settings"
- Select Change Advanced Power Settings
- Go to Sleep section
- Find "Allow Wake Timers" and enable it

4.10.4 CPU Usage Alert

It was also suggested that YouInteract should be able to evaluate CPU processing. In order to achieve this, while the system is running, there is a thread specifically tracking the CPU usage. It read its value every second and compares that value with value from the previous second. If the percentage usage is higher than 90% for 20 seconds, then it is registered in the log file the date time it occurred. If the system administrator wants, every time this alerts occurs, in addition to register it in the log file, YouInteract may restart the connection between Microsoft Kinect and YouInteract since most of performance issues are related to the data obtained by Microsoft Kinect.

Chapter 5

YouInteract:Applications

YouInteract was designed to support applications within its system. These applications can be either developed in a WPF context or be a game engine type of application. The fact that YouInteract is able to support applications, created with a game engine, facilitated the development of new 3D applications for FCCVA. This chapter focus on the tools developed to help developers create new YouInteract Applications, why and how the game engine Unity3D was chosen to create new applications and a description of the applications developed during this project's time.

5.1 Unity 3D

In addition to supporting WPF applications, one of the requirements of FCCVA was that YouInteract needed to be able to run more demanding applications such as 3D engines so it was decided to support applications created with the game engine Unity 3D. The integration of this type of files applications is similar to WPF applications. There is a directory responsible for having the executable and data files for each application and the YouInteract system will initialize them if the configuration file has the referred App active. There are some recommendations and guidelines a developer of a Unity application for YouInteract should follow, for example, the name of the application must always start with "U_", indicating this is an application made with Unity for YouInteract (see Appendix E for more details).

As figure 5.1 shows, when the user clicks in a Unity App icon, the system immediately stops the connection between the Microsoft Kinect and the YouInteract and shuts down its sensor. This is needed since the Unity application is independent from WPF and has its own wrappers to access data from this hardware. Since this strategy takes five to six seconds to be completed, a loading page appears in the display showing the user that the system is loading with a description of the application chosen to run.

On the other side, immediately after the user clicks in the appropriate button to exit that Unity3D application, a loading page is again displayed while the connection between Microsoft Kinect and the YouInteract system is restored. After completing that task, the user is now able to navigate through the main Menu of YouInteract.

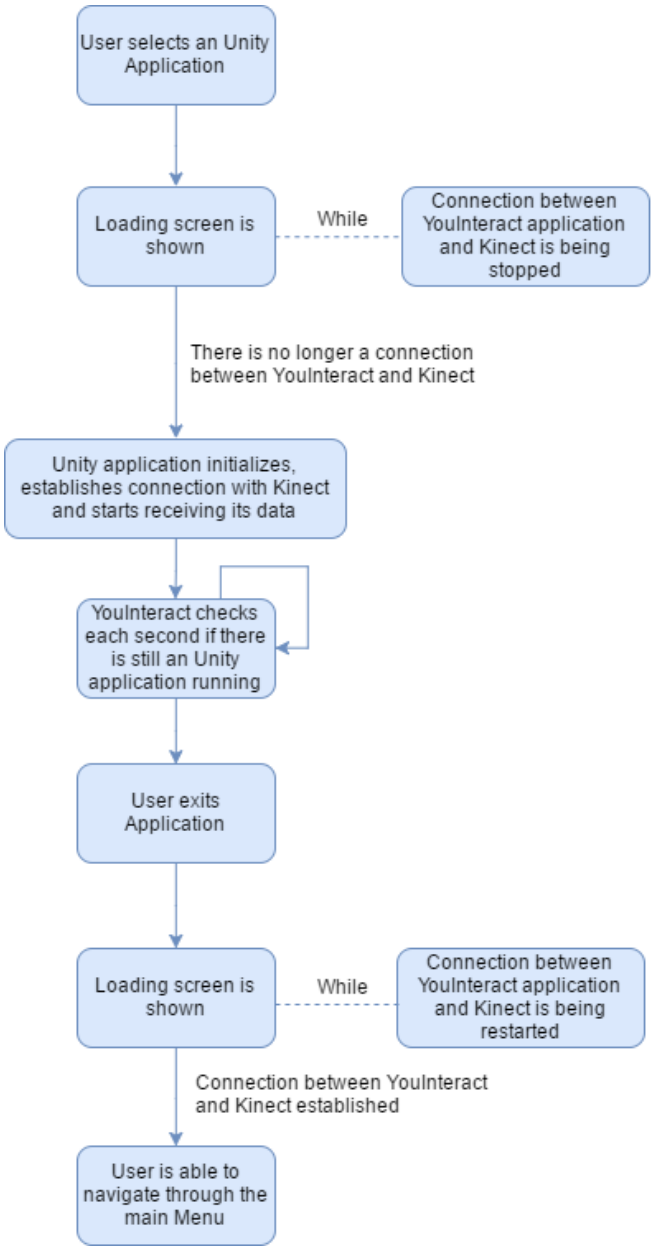


Figure 5.1: Flow diagram of what happens when an Unity application is launched.

5.2 YouInteract API

In order to allow students to develop new applications for YouInteract, a YouInteract API was developed. It provides a set of methods responsible for handling several aspects that an applications needs to have in order to properly function in the YouInteract system. This group of functions and controls allow to hide some of the complexity behind YouInteract, it was named YouInteract API and it is a DLL file that must be added as a reference in your application's project as referenced in the manual [see Appendix C].

With this API, the developer does not need to be concerned about many implementation details:

- Initialize and deal with Microsoft Kinect
- System navigation.
  - Navigation inside the application.
  - Navigation to Main Menu.
  - Release of resources not needed anymore.
- Two types of buttons the user can select.
- Scroll arrows for simpler navigation.
- User viewer.
- Screen's height and width.
- Application's log file creation.
- Directory's paths to retrieve specific images, themes, icons, configurations...

YouInteract API rests on a combination of functions provided by Microsoft Kinect SDK that were changed and modified and adapted to YouInteract specifically. By using them, the developer is abstracted from the complexity present(Figure 5.2).

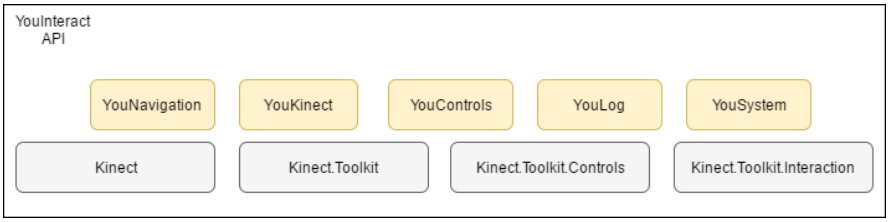


Figure 5.2: Blocks that make the YouInteract API.

There are several functions available to the developer. The detailed explanation about each one of the functions is available in the documentation (Appendix C) and an explanation and samples on how to use it are given in the manual for developing new Applications (Appendix D).

5.3 Apps

This version of YouInteract already comes with a group of applications ready to be used. Four of them are WPF applications (see Appendix D for how to develop a WPF application for YouInteract), created in previous years while one WPF application and five Unity 3D applications were created for this dissertation (see Appendix E for how to develop a Unity application for YouInteract).

5.3.1 Existing Apps

. The following applications already existed in previous versions of YouInteract and were modified in order to be in accordance with the developed YouInteract API as well as changes on the memory management since none of them were concerned about the high use of resources specially when it came to render images.

You_Gallery

The goal of this application is to show a list of images that were uploaded and chosen through the Portal (Figure 5.3). Comparing to the previous version, the main changes were in the interface and the memory management of the images shown.

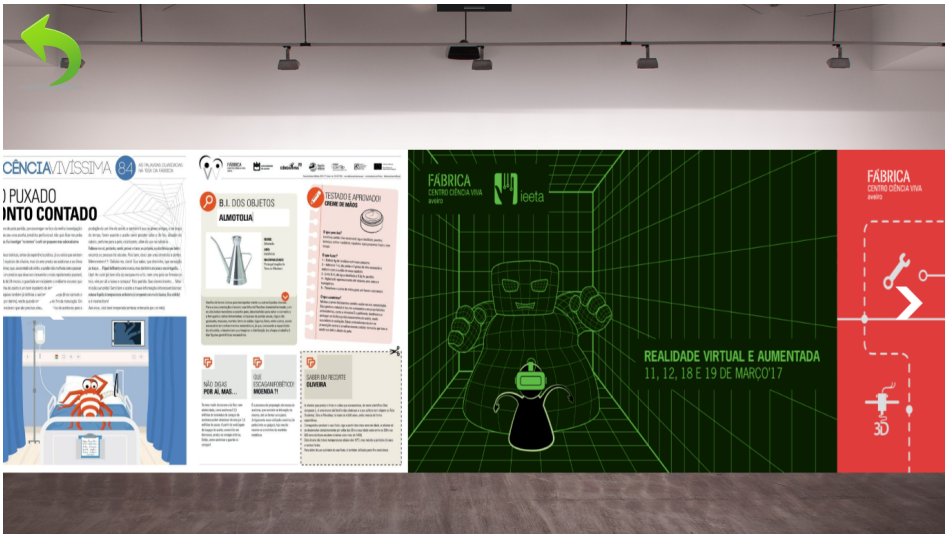


Figure 5.3: You_Gallery view shown to the user.

You_Videos

The goal of this application is to show a list of videos that were uploaded and chosen through the Portal. After the user selects one of them, it is played in full screen offering a chance to the user stop, pause and play it again by clicking in the buttons (Figure 5.4). Comparing to the previous version, it suffered major changes in the structure because it was outdated. Since it used tools no longer supported in the last .Net Framework versions, the idea stayed the same but the implementation was changed is several aspects.

Figure 5.4: You_Videos views shown to the user.

You_Weather

The goal of this application is to show the weather for the next days to a specific location (Figure 5.5). Comparing to the previous version, it only suffered changes to support the YouInteract API and in the basic structure to be in accordance of the rules established (Appendix D).

Figure 5.5: You_Weather views shown to the user.

You_TicTacToe

The goal of this application is to offer the user a chance to defeat the ”computer” in a tic-tac-toe game (Figure 5.6). This was an application that already existed in previous versions however its architecture was too complex and confusing and had features that were no longer needed. So it was redesigned taking advantage of the existing visual interface and its game logic.

Figure 5.6: You_TicTacToe views shown to the user.

You_Slideshow

The goal of this application is to display a slide show of images that were chosen through the Portal (Figure 5.7).

Figure 5.7: You_Slideshow view shown to the user.

5.3.2 New Apps

In addition to the previous applications, since YouInteract also supports Unity, new applications were developed. Although the following applications may be used in other locations aside from FCCVA, these applications were created and designed according to the FCCVA requirements.

An asset available at the Unity store, for free, was used in order to simplify the integration of Kinect with Unity (Kinect with MS-SDK from RF solutions).

U_AvatarMovement

This Unity application was developed specifically for the first day of tests (Chapter 6) and shows a virtual avatar mimic the movements of the user. The users are able to see their movements in a virtual avatar (Figure 5.8) or in a skeleton avatar. In order to select the method desired, the user can select the chosen button in the upper right corner. The application is also able to detect two users at the same.

Figure 5.8: U_AvatarMovement virtual environment that is shown to the user.

U_BallsFalling

This Unity application was also developed specifically for the first day of tests (Chapter 6). It is an augmented reality application since it shows the image of the real users and overlaps a group of virtual balls which the user can interact with (Figure 5.9).

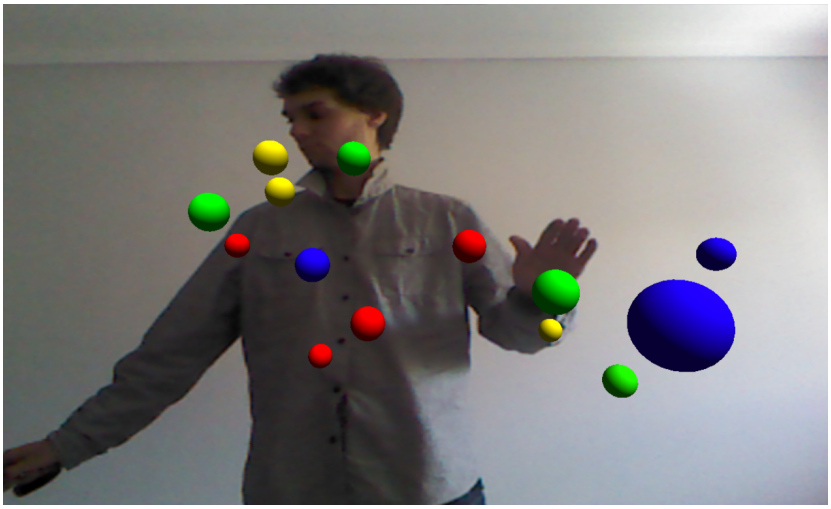


Figure 5.9: The game is a overlap of virtual 3D balls with the real RGB image captured by Kinect.

U_Football

This Unity application is an evolution of the application U_AvatarMovement and consists in an avatar that mimics the users movements where the objective is to kick a virtual ball and score a goal. The application is able to check the impact position of the user's foot and the ball in order to evaluate the trajectory and velocity the ball gains. It has five different levels where in each level obstacles appear in front of the football net to difficult the user's goal. The user has three attempts to score the maximum goals possible (Figure 5.10).

Figure 5.10: Image shown to the user when he is playing the U_Football game.

U_Bubbles

This Unity application is an evolution of the application U_BallsFalling. It still consists in an augmented reality application with virtual bubbles being shown over the user's real image. Its main goal is to pop the maximum bubbles possible in forty-five seconds (Figure 5.11).

Figure 5.11: U_Bubbles is a game that overlaps virtual bubbles with the real RGB image captured by Kinect.

U_UANavigation

This Unity application is a recreation of an application developed in 2015 by João Cardoso [Cardoso, 2015] and offers the user the possibility to navigate through a virtual model of the University of Aveiro. Since the application was created using NeoAxis 3D Engine, which YouInteract does not support, an Unity application similar to the one created by João Cardoso was created. It was possible to render the campus of University of Aveiro using the existing .obj files from the previous work.

The manipulation of 3D objects using gestures is being studied for a long time, as [Bowman and Hodges, 1997] mention this capability as a desirable feature in many VR applications. This application offers the user a chance to manipulate a virtual steering wheel. It evaluates user's both hands position to control the virtual steering wheel (Figure 5.12).

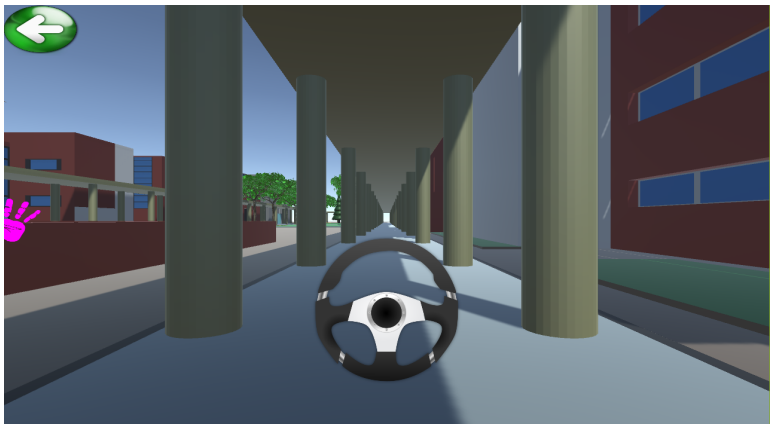


Figure 5.12: U_UANavigation shows a 3D model of University of Aveiro where the can control a virtual steering wheel to navigate through the model.

Chapter 6

YouInteract: Tests

This chapter describes the two days of testing YouInteract in a public environment. The tests were made in Fábrica Centro Ciência Viva de Aveiro and every participant user was either a visitor or a person who worked there. In the first day of testing the main goal was to confirm the system's robustness on working several hours straight with several people interacting with it and to observe what was the most important issues that could be improved. After analyzing the collected information, the system suffered some changes and was put to a second day of tests also in Fábrica Centro Ciência Viva de Aveiro.

6.1 Setup

In both days of tests, YouInteract application was ran in a laptop with Windows10 as operative system, a Intel Core i5-4210U processor, 6 Gb of RAM and a Nvidia Geforce 840M as main hardware components. It had a television as display and used as input a Microsoft Kinect v1.

The setup was located in an hallway where the users had plenty of space to be comfortable in interacting with the system (Figure 6.1).

Before each tests it was advised that the Internet connection signal at the place where the setup would be installed was extremely weak so the configuration was done previously in order to not depend on the internet signal to connect to the YouInteract portal.

Figure 6.1: Location of where the setup was installed.

6.2 First day of tests

On the first day of testing the system, 12th April 2017, the target audience to use YouInteract were subjects of both genders between 6 and 12 years old without any physical restriction that were visiting Fábrica Centro Ciência Viva de Aveiro.

The tests were programmed to be divided in two parts, one from 11:00 to 12:00 and the other from 14:00 to 15:00 however after analyzing the log data of the day, it was concluded that the real function hours of the system were from 10:32:41 to 11:52:00 in the first half and from 13:31:48 to 14:51:47 in the second half.

During these hours, 16 participants interacted with the system. They came in groups of four and three of them stayed behind the person that was in a central position interacting with the system (Figure 6.2).

The Main goals of the tests were:

- Verify the robustness and functioning of the YouInteract system by analyzing the number of crashes and possible slowness in the tasks processing.
- Analyze the interaction with the system looking for possible types of problems on how to understand and how to use the application.
- Check what are the most used applications.
- Ask the participants for suggestions of future applications they would like to see in next versions.

Figure 6.2: Different groups of participants interacting with YouInteract on the first day of tests.

It was given all liberty to the participants to navigate through the YouInteract and in the beginning no instructions on how to use the system were given to the participants, in

order to check if it was intuitive enough. Explanations were only given when it was observed interacting problems or when the participant asked for help.

At the time of the test the available applications were:

- You_Videos
- You_TicTacToe
- You_Gallery
- You_Weather
- U_AvatarMovement
- U_BallsFalling

Figure 6.3 shows the usage percentage of the applications. As expected the most used applications were the games created with Unity3D and the TicTacToe application in the third place. The other applications were almost ignored and did not aroused interest in the target audience.

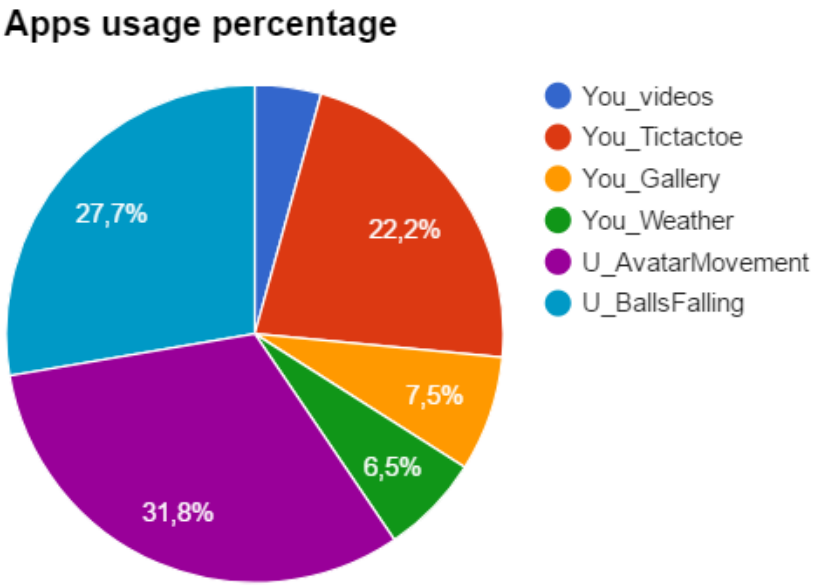


Figure 6.3: Graphic showing the application's usage percentage of the first day of tests.

Since the participants were very young, we decided not to use a formal questionnaire but rather to observe their behaviour and ask informal questions about their thoughts and suggestions.

The participants understood they had to use hand gestures in order to interact with the system but they had some difficulty understanding the movements they needed to do. Most of them started with the stretched arm so they could not perform the press movement to click in the buttons which led to the development of a Timer Click in order to avoid this (see

Section 4.4.1). After an explanation and a few tries, most of them understood and used the system with no further problems. They understood how the application worked and navigated between YouInteract's Apps with no problems. Since the participants were in groups of four, it was difficult to restrain the space to only one of them so the more time passed, the more occurrences of them trying to interact with the system at the same time happened (Figure 6.2).

Conclusions of the observation of the most used applications:

- You_TicTacToe - It was the third most used applications and the participants had no problems playing with it. They gave a suggestion on having a mode to play against another person since the applications only allowed to play against the computer.

- U_BallsFalling - It was used with great enthusiasm by the participants since it was an application different from what they are used to because it is part of a Augmented Reality context and the participants liked the fact that they could see themselves interacting with virtual balls. However several problems were detected, regarding the difficulty of the participants to understand the level of depth of the virtual balls related to their real body which led to difficulties interacting with the virtual balls. In addition to this, since their image was shown in the television it led all of the participants to try playing the game at the same time.

- U_AvatarMovement - It was also played with great enthusiasm since this application supported two people at the same time. The participants liked to see their movements replicated by the virtual avatars.

It is also important to refer that the Unity applications took a few seconds to initialize and no feedback was given to the user indicating the system was loading. It also was concluded that most of the participants wanted the games to have an objective and U_BallsFalling and U_AvatarMovement failed this standard.

As a conclusion of the tests and after analyzing the log file of the day, it was concluded that the system did not have any performance problems and dealt the interaction with no problems. It was also noted the majority of participants liked and used the system with great enthusiasm although having some problems understanding on how to use it the first time since they started with the arm lifted and did not understood very well how the buttons were pressed.

6.3 Second day of tests

On the second day of tests, 10th May 2017, the target audience to use YouInteract were subjects of both genders between 14 and 15 years old.

The system was on and ready to use from 10:06:26 to 13:02:36 and 20 users interacted with it. Like the first test, they came in groups but everyone stayed behind the person that was using YouInteract in order to not disturb the user's navigation (Figure 6.4).

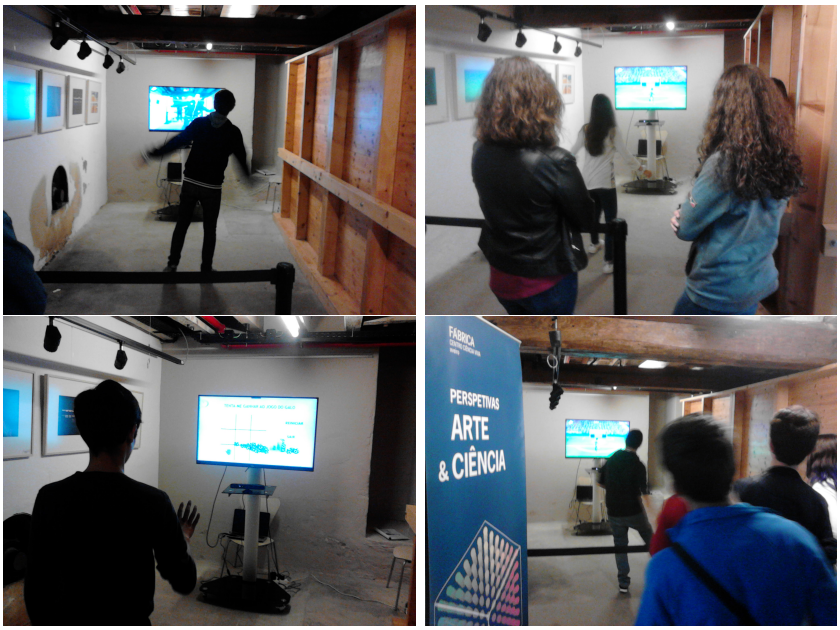


Figure 6.4: Different groups of participants interacting with YouInteract on the second day of tests.

The differences between this and the first test were mainly the three new applications created to catch the users attention, some bugs were corrected and a feedback every time a Unity3D application was launched appeared. With this in mind, the main goals of this second test were:

- Continue to verify the robustness of the YouInteract system by analyzing the number of crashes and possible slowness in the tasks processing.
- Analyze the interaction with the system looking for possible types of problems on how to understand and how to use the application.
- Do a small questionnaire to every participant in order to understand key elements about the satisfaction in using YouInteract.

As in the first test, the participants could use the application with no guidance and navigate through the YouInteract (Figure 6.4). Explanations were only given when problems occurred or when the participant asked for help. From the applications developed during this project's time (see Chapter 5), at the time of the test the available applications were:

- You_Videos

- You_TicTacToe
- You_Gallery
- You_Weather
- U_Bubbles
- U_Football
- U_UANAvigation

Figure 6.5 shows the usage percentage of the applications and the most used applications were U_Bubbles, U_Football and U_UANAvigation, applications created with Unity3D.

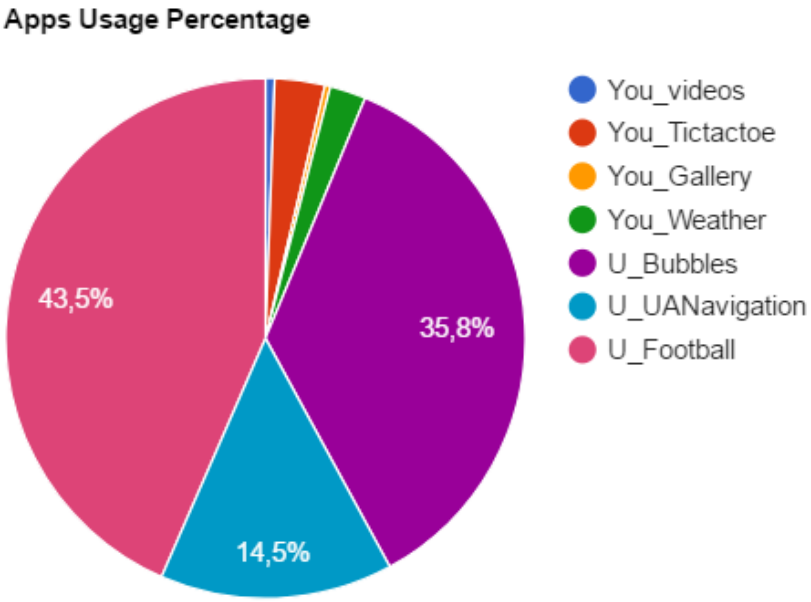


Figure 6.5: Graphic showing the application's usage percentage of the second day of tests.

At the end of each participant session, a small questionnaire with five direct questions was given:

- How much did you like the Applications? Please rate from 1 to 5, being 1 did not like at all and 5 liked it very much.
- Was it easy to understand how to navigate through the Applications? Please rate from 1 to 5, being 1 very difficult and 5 very easy.
- How easy was to use the system? Please rate from 1 to 5, being 1 very difficult and 5 very easy.
- Which application did you like the most?
- Do the applications have annoying characteristics? Please rate from 1 to 5, being 1 did not have annoying characteristics and 5 a lot of annoying characteristics.

Figure 6.6 shows the different results to each of the questions of the questionnaire that was made to the participants.

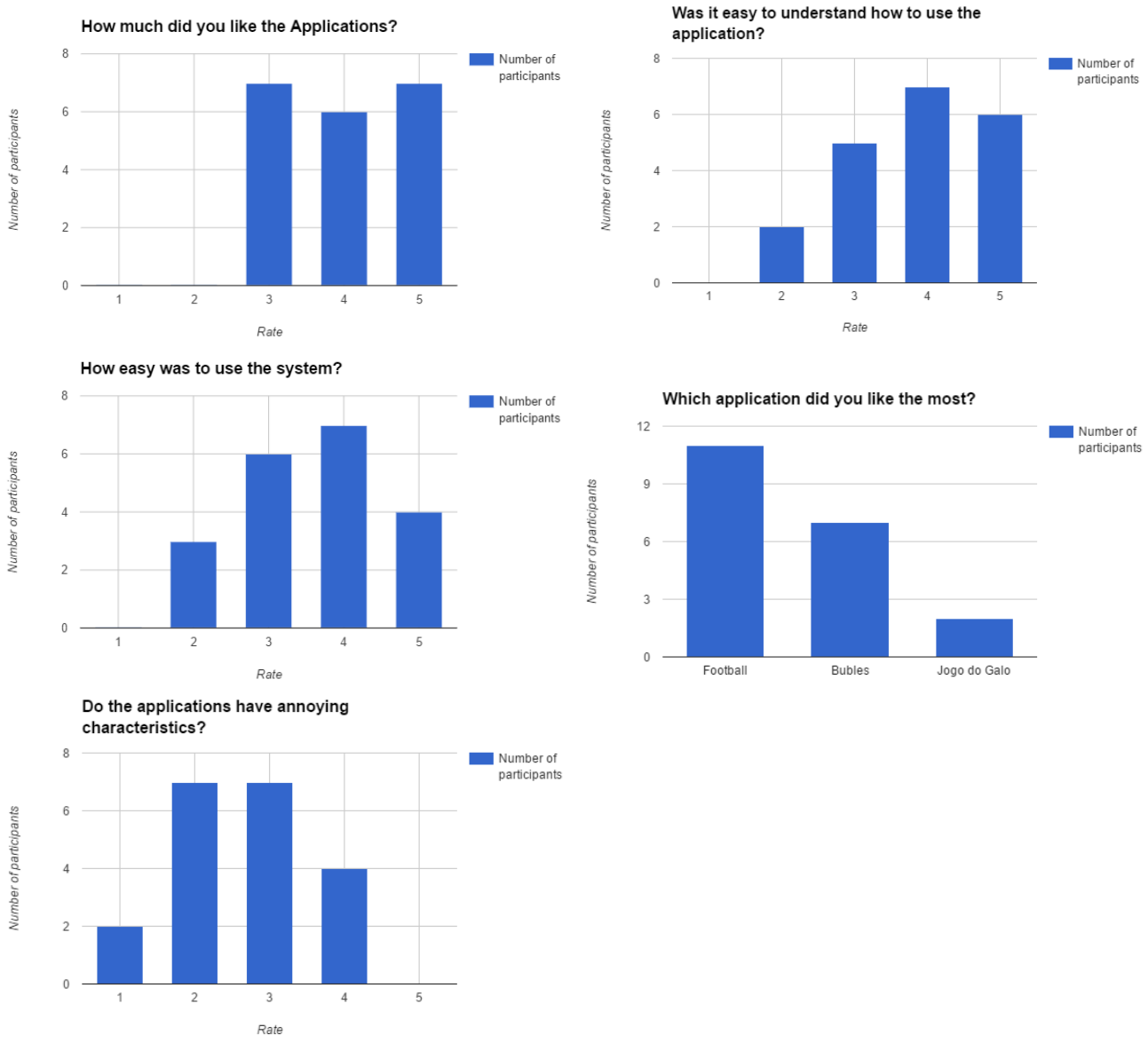


Figure 6.6: Graphics showing the results for the five different questions of the questionnaire.

After analyzing these results, it is possible to conclude that the available applications are liked by most of the participants and the only suggestion made was that YouInteract should have more available Apps, specifically games. Most of the users understood how the system worked however they had some difficulty in the first seconds since it was not given any initial explanation on, for example, how to click the buttons. The most liked applications were U_Football and U_Bubbles however participants found some inconsistencies in clicking the buttons in the main menu of YouInteract and clicking the buttons on these Unity3D Apps since the system used to click the buttons is different, Timer Click vs Push Click (see Chapter 5).

As a conclusion of the second day of tests and after analyzing the log file of the day, it was concluded that the system did not have any performance problems and dealt the interaction with no problems. The majority of participants liked the available applications but suggested to add some more. However, it was observed a major problem since the participants have difficulty on understanding how the system works and need a brief explanation to start interacting with it the right way. It is also important to standardize the way the buttons are clicked so the way of clicking in them is the same in every part of YouInteract.

Chapter 7

Conclusion and Future Work

The main objective of this dissertation was to improve an interactive public display system under development at the University of Aveiro. One of the objectives was to develop a system flexible enough to adapt easily to different context by providing personalized content to the users through a web portal. The system should be robust and able to run for several hours a day without performance issues. Similar projects using gesture recognition exist however most of them are limited to a single configuration whereas the system we present is flexible enough to allow easy configuration of running applications and contents.

The architecture was selected to allow easy and remote configuration. Care was also taken to ensure robustness: the system can run for several hours every day since proper resources management was handled. After the detection of a user, the system changes its state and is able to recognize gestures allowing the user to navigate through a menu containing several different applications. The main application was developed to make possible an easy integration of other applications created by other developers. An API was specifically created to ease the development of ther applications for YouInteract.

In addition to the original WPF applications, the system was extended to support 3D applications created with Unity 3D. The idea was to give future developers the possibility to create complex and immersive 3D applications that still use Microsoft Kinect for gesture recognition. This was motivated by our pilot study in Fábrica Centro Ciência Viva de Aveiro, since 3D games where particularly interesting for their attendance.

Since the system is supposed to be installed in a public space for several hours, a mechanism was developed to allow the administrator to check what happened during working hours. This LOG system records events while YouInteract is running such as the applications usage, the number of people that interacted with the system, performance problems, among others.

Besides the main application, an existing web portal was further developed to configure easily and remotely the different instances of the application running. The website that communicates with the application was adapted from previous work where a system administrator would have the possibility to manage and configure each device running the application created. The main goal of this web site is to provide a way of changing the contents available in a certain display through the Internet and not having the need to access physically each machine.

To validate the system created, user experiments were performed in a public space. The system was installed in Fábrica Centro Ciência Viva de Aveiro in order to evaluate its performance as well as to observe how users interacted with it. The actions while interacting

with the system were observed in order to understand what could be improved and several suggestions were given by the users. Based on that information the system suffered some changes with the insertion of a Timer Click that helped to overcome the button selection problems observed. A tutorial video, explaining how to interact with the system, was also added that starts when the system detects a user not making any action for a given time.

The results suggested the system created is robust enough to be running several hours a day without any performance issue. It is currently installed at the Department of Electronics, Telecommunications and Informatics of University of Aveiro and is ready to be installed permanently in other public spaces. During the test, most of the users enjoyed interacting with the system and many suggest that YouInteract should integrate more applications.

This dissertation extends our knowledge concerning how to take advantage of affordable displays and gesture-tracking hardware to make a personalized interactive system that can be installed in innumerable public spaces.

Future work should focus on optimizing some of the gestures recognition in order to improve the navigation through the system. It would also be relevant to study and implement new mechanisms that would capture the people attention and give some hints on how to use it (the video developed in the late part of this thesis is a first attempt towards this direction). Finally, the system is open and thus it would be interesting to develop other interesting application for entertainment or not, taking advantage of the API developed within this thesis.

References

Boutsika, E. Kinect in education: A proposal for children with autism. In *Procedia Computer Science*, volume 27, pages 123–129, 2013. doi: 10.1016/j.procs.2014.02.015.

Bowman, D. A. 3d user interfaces, 2014. In Soegaard, M. and Dam, R. F., editors, The Encyclopedia of Human-Computer Interaction, chapter 32. Aarhus, Denmark: The Interaction Design Foundation, 2nd edition.

Bowman, D. A. and Hodges, L. F. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments, 1997. In Proceedings of the 1997 Symposium on Interactive 3D Graphics, I3D '97, pages 35–ff., New York, NY, USA. ACM.

Bowman, D. A., Kruijff, E., LaViola, J., and Poupyrev, I. 3d user interfaces: theory and practice, 2004. URL `http://www.lavoisier.fr/livre/notice.asp?id=O3OWRLAASXLOWK`.

Cardoso, J. A. C. 3d manipulation and navigation methods with gestures for large displays, 2015. Masther's thesis, University of Aveiro.

Cassola, F., Morgado, L., de Carvalho, F., Paredes, H., Fonseca, B., and Martins, P. OnlineGym: A 3D Virtual Gymnasium Using Kinect Interaction. *Procedia Technology*, 13:130–138, 2014. doi: 10.1016/j.protcy.2014.02.017.

Dias, P., Sousa, T., Parracho, J., Cardoso, I., Monteiro, A., and Santos, B. S. Student projects involving novel interaction with large displays, 2014. University of Aveiro.

Duarte, F. M. M. Interacção com ecrãs públicos através de um dispositivo móvel, 2011. Masther's thesis, University of Aveiro.

Garber, L. Gestural Technology: Moving Interfaces in a New Direction [Technology News]. *Computer*, 46(10):22–25, 2013. doi: 10.1109/MC.2013.352.

Hinrichs, U., Carpendale, S., Valkanova, N., Kuikkaniemi, K., Jacucci, G., and Vande Moere, A. Interactive Public Displays. *IEEE Computer Graphics and Applications*, 33(2):25–27, 2013. doi: 10.1109/MCG.2013.28.

Hsu, H.-m. J. The potential of Kinect in Education. *International Journal of Information and Education Technology*, 1(5):365–370, 2011. doi: 10.7763/IJIET.2011.V1.59.

Laravel. Laravel. `https://laravel.com/`. [Online, Access: 09/06/2017].

Laravel-blade. Blade. `https://laravel.com/docs/5.3/blade`. [Online, Access: 09/06/2017].

Laravel-database. Database. https://laravel.com/docs/5.3/database. [Online, Access: 09/06/2017].

Laravel-eloquent. Eloquent ORM. https://laravel.com/docs/5.3/eloquent. [Online, Access: 09/06/2017].

Laravel-migrations. Database: Migrations. https://laravel.com/docs/5.3/migrations. [Online, Access: 09/06/2017].

Laravel-queries. Database: Query Builder. https://laravel.com/docs/5.3/queries. [Online, Access: 09/06/2017].

Laravel-seeders. Database: Seeders. https://laravel.com/docs/5.3/seeding. [Online, Access: 09/06/2017].

Motta, T. and Nedel, L. Interaction with public displays using a natural user interface based on an extended version of kinect sdk, 2013. Institute of Informatics Federal University of Rio Grande do Sul.

MySQL. MySQL. http://www.mysql.com/. [Online, Access: 09/06/2017].

OS-Marketshare. Most used operative system. https://www.netmarketshare.com/operating-system-mark [Online, Access: 09/06/2017].

Palha, R. DetiGuide: Interagindo com um ecrã de grandes dimensões através de um dispositivo móvel, 2012. Masther's thesis, University of Aveiro.

Parracho, J. Manipulação de objetos e navegação em ambientes virtuais com o kinect, 2013. Masther's thesis, University of Aveiro.

PHP. PHP FAQ. http://php.net/manual/en/faq.general.php. [Online, Access: 09/06/2017].

Reflection, C. C# reflection. https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/c [Online, Access: 09/06/2017].

v1 requirements, K. Kinect v1 requirements. https://msdn.microsoft.com/en-us/library/hh855359.aspx [Online, Access: 09/06/2017].

v2 requirements, K. Kinect v2 requirements. https://msdn.microsoft.com/en-us/library/dn782036.aspx [Online, Access: 09/06/2017].

Walter, R., Bailly, G., Valkanova, N., and Muller, J. Cuenesics: Using mid-air gestures to select items on interactive public displays, 2014.

Zhang, Z. *Microsoft kinect sensor and its effect.* 2012. doi: 10.1109/MMUL.2012.24.

Appendix A

YouInteract Portal installation tutorial

The computer running the YouInteract Portal needs to have the following software installed:

- Apache2.
- PHP 7.*.
- OpenSSL, PDO, Mbstring, Tokenizer, XML PHP Extensions.
- MySql.
- Composer.
- Laravel.

The following steps are a recommendation on how to install all the software needed to run the Portal.

1. Install WampServer - http://www.wampserver.com/en/

   WampServer is a Windows web development environment that installs automatically Apache2, PHP, PHP extensions and MySql.

2. Install Composer - https://getcomposer.org/

   Laravel utilizes Composer to manage its dependencies. So, before using Laravel, the machine must have Composer installed.

3. Install Laravel - https://laravel.com/docs/5.4server-requirements

   Open the Windows command line and insert ≪composer global require ”laravel/installer”≫.

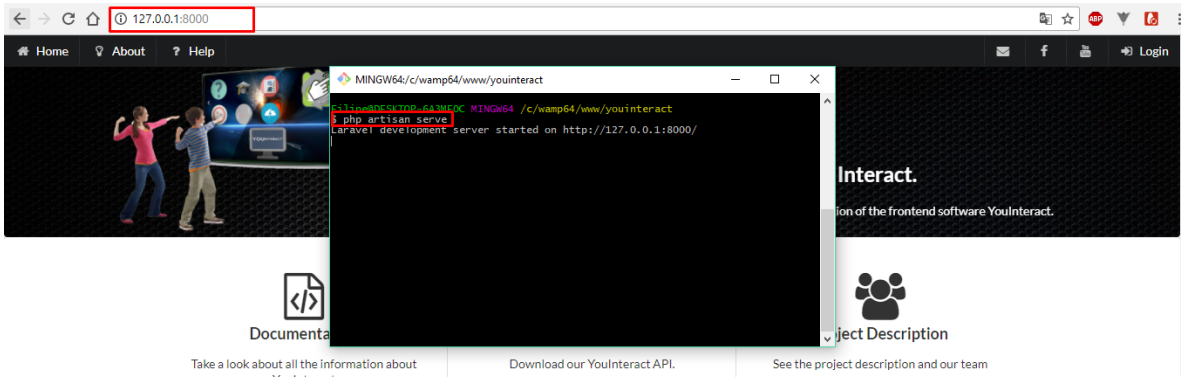
4. Run Portal

   (a) Go to the directory chosen to install WampServer (usually ”C:/Program Files/wamp64” and copy the existing files of YouInteract Portal to the folder ”www”.

   (b) In the Windows command line, go to the location of the Portal files (”C:/Program Files/wamp64/www/youinteract”) and insert ”php artisan migrate” in order to set up the database.

   (c) Then, insert ”php artisan db:seed” to populate the database.

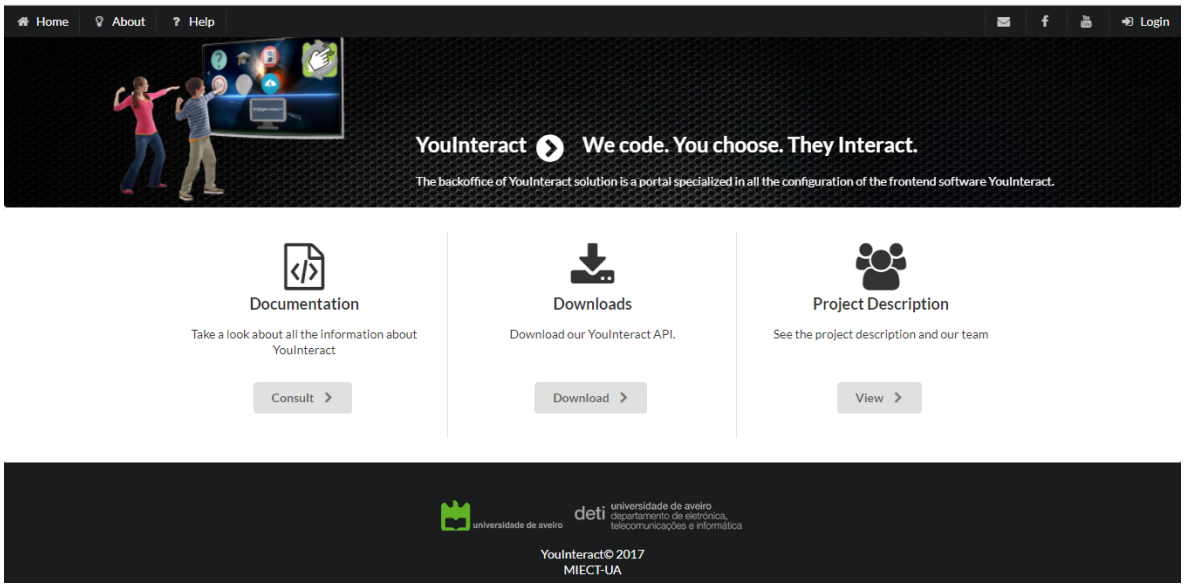
   (d) Finally insert ”php artisan serve”. The Portal can be accessed inserting the link showed on the console.



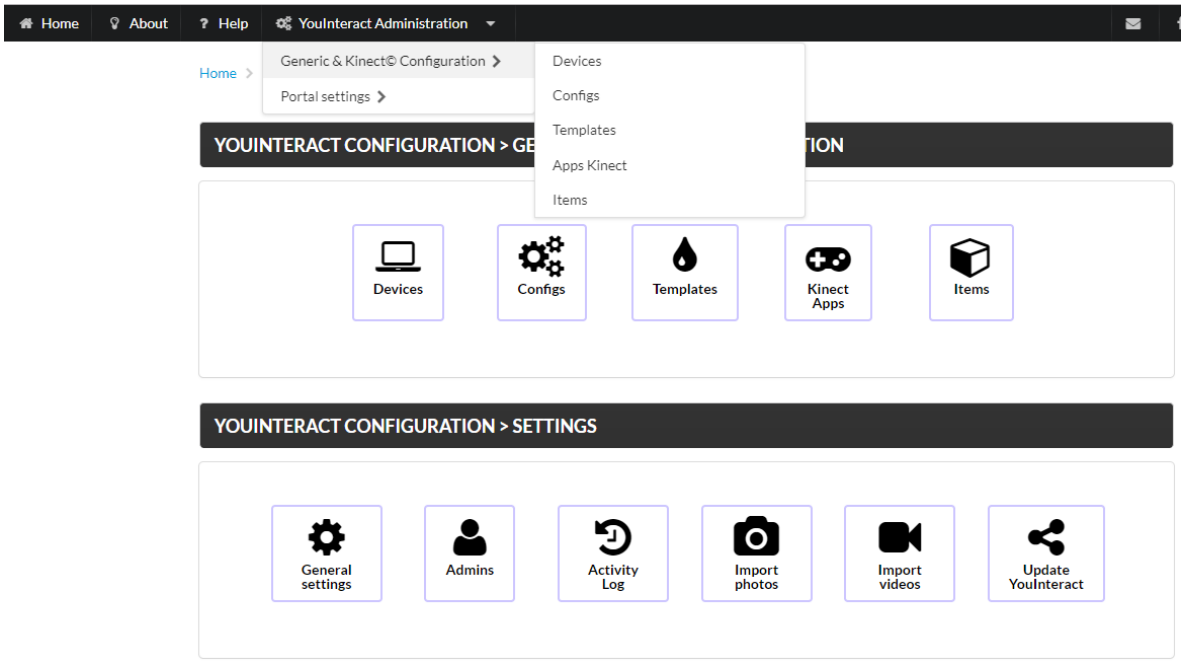
Appendix B

YouInteract Portal manual

The home page of YouInteract Portal allows the visitor to consult the YouInteract documentation, download its API and check the project's description. In the right upper corner it also has the button to login.

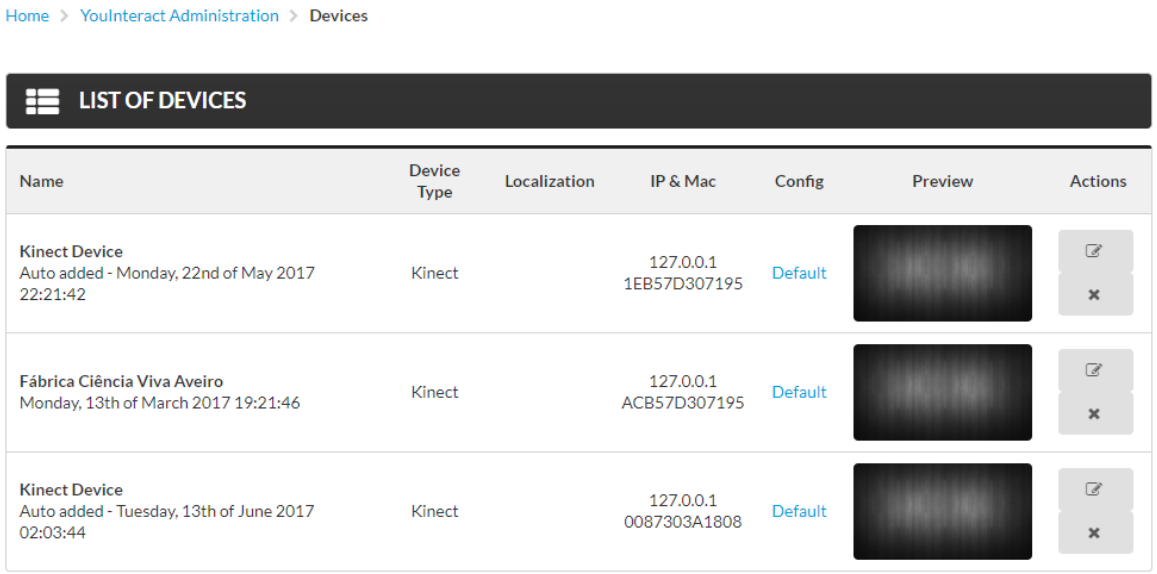


After making the login, it is presented the main board where most important features of the website are accessible in one click. The administrator can access the different parts of the Portal using the icons or by accessing the toolbar that will remain in the same location in every page of the website.

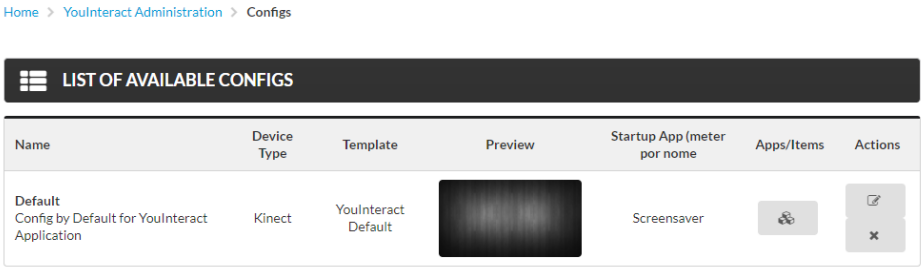


By going to "Devices", it shows a list of every machine that has YouInteract installed and ran it at least one time. It is possible to know if the machine has a Kinect connected, its IP and MAC addresses, the time it last communicated with the Portal and its configuration.

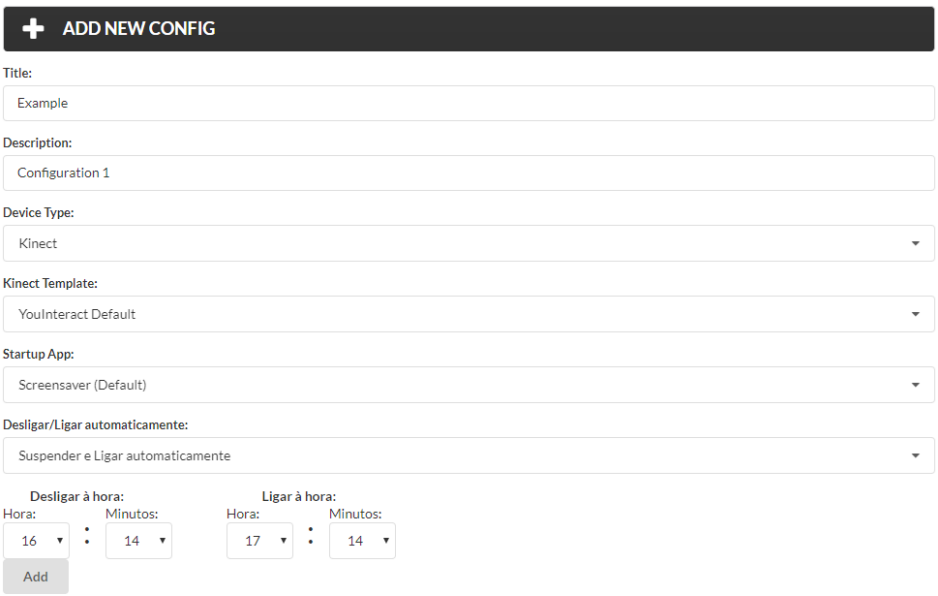
On the "Actions" column it is possible to delete a device or change its information and configuration.



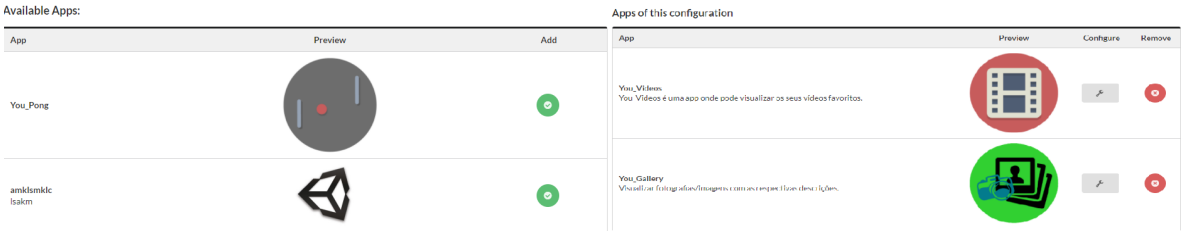
By goind to ”Configs”, it shows the description of every configuration created. It shows the configuration name, the template used with a preview of the background image and the application that YouInteract will start as soon as it initializes.



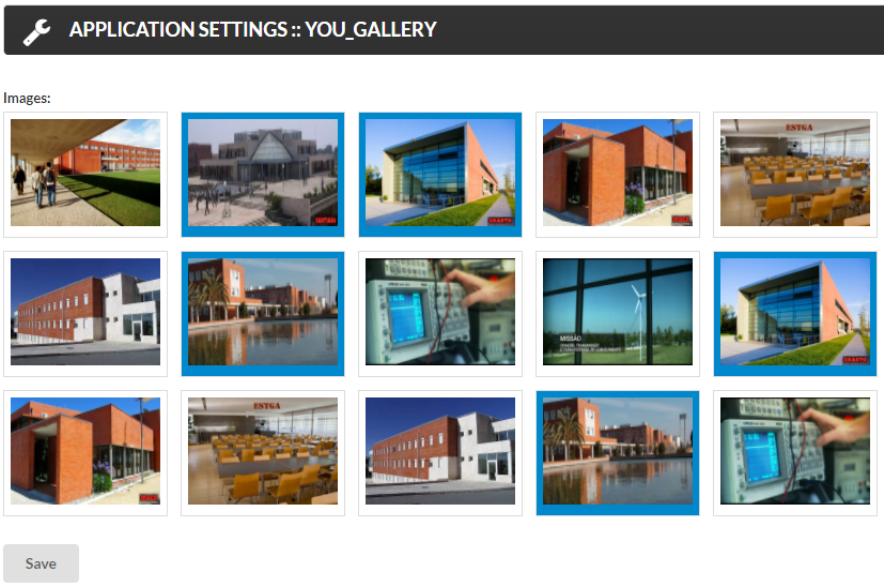
This page also allows to add a new configuration with the fields chosen by the administrator.



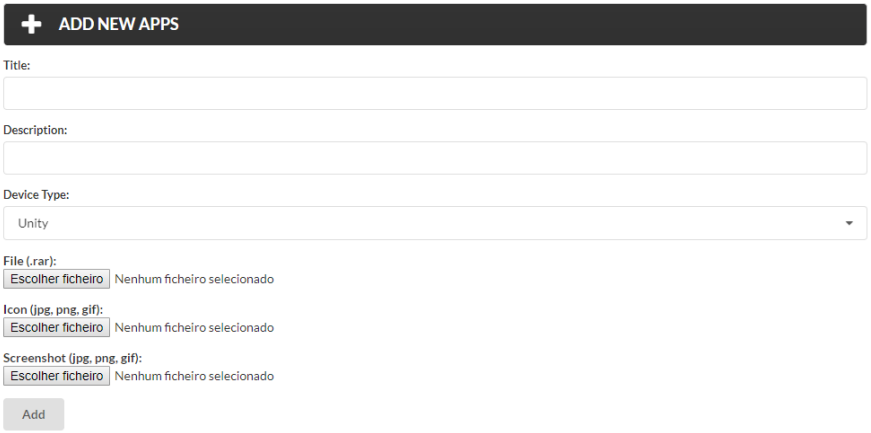
By clicking on the ”Apps” column of a configuration, the administrator can select which applications will be available in the YouInteract with that configuration. The applications that are already in the configuration are shown first and can be removed by clicking in the ”X” icon while the applications that are still not selected can be added by clicking in the ”tick”.



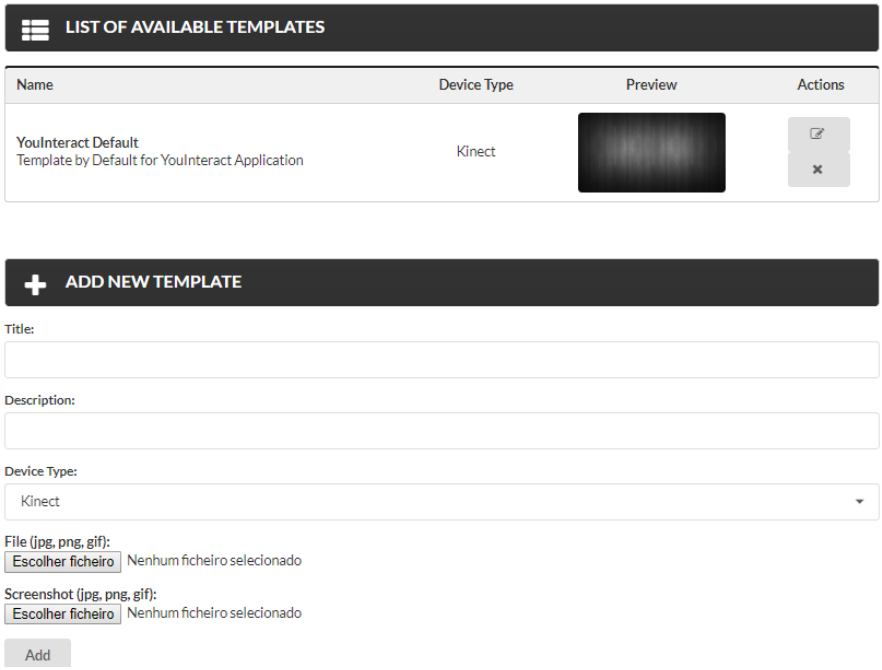
Some of the applications can be configured (for now just You_Gallery and You_Videos). The administrator can select which images/videos will be available in the application.



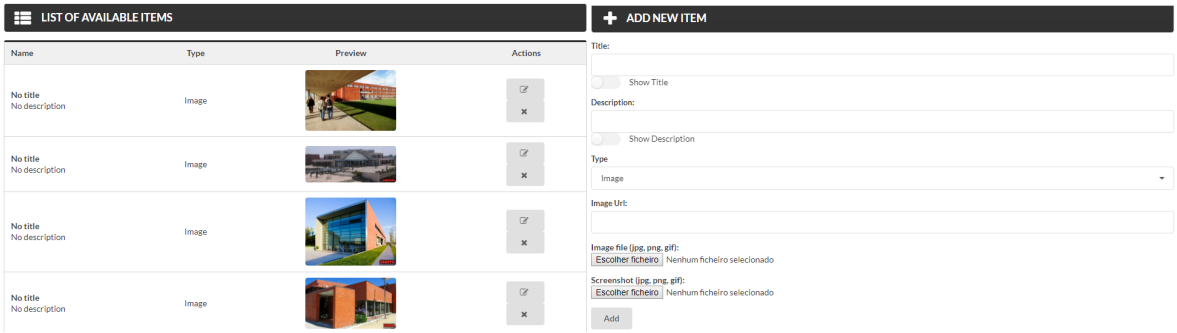
It is possible to add new applications where the administrator must indicate what type of application wanted to add: WPF or Unity application.



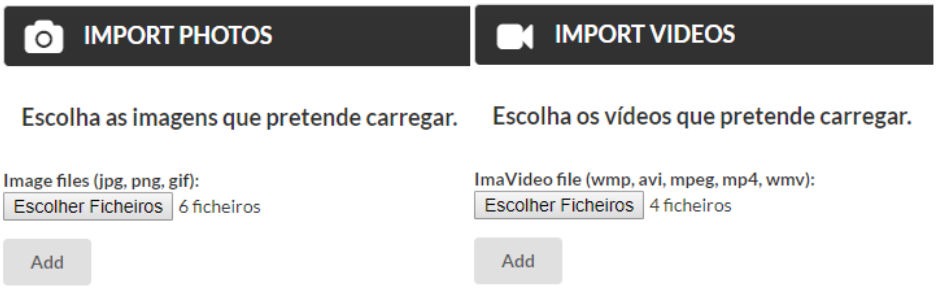
By going to "Templates", the administrator can check all the existing templates as well as remove or change their information. It is also possible to add new templates by uploading an image file, the name and description.



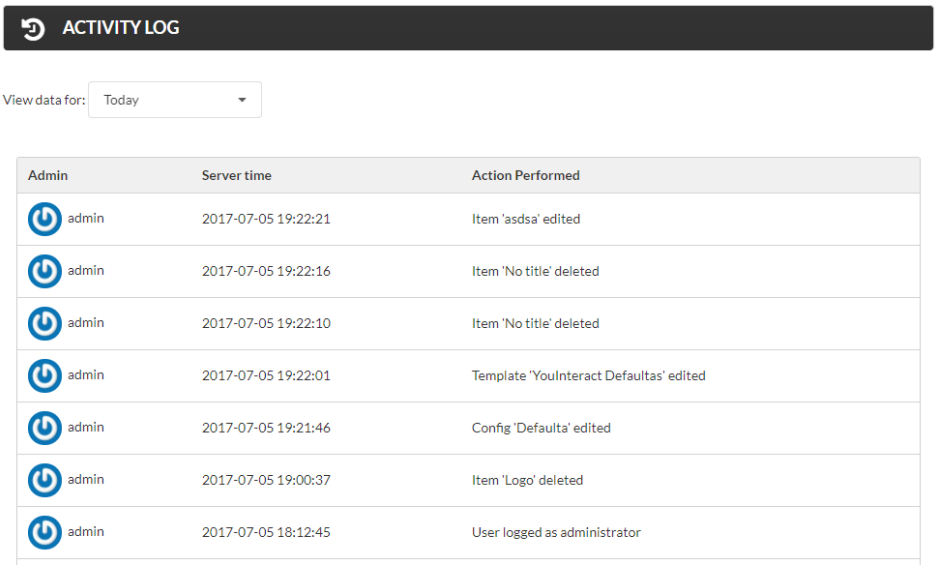
By going to "Items", it is possible to check all the existing images and videos as well as removing them or change their information. It is also possible to upload new ones (one item at a time).



If several images or videos are needed to be uploaded, the administrator can access "Import Photos" or "Import Videos" and upload several files at the same time.



To check every thing that was done in the Portal, an activity log can be accessed.



Appendix C

YouInteract API documentation

YouInteract API can be divided in five parts: YouSystem, YouKinect, YouNavigation, YouControls, YouLog.

C.1 YouSystem

```
/* HandTimer */

//Event raised when virtual hand is over a YouButton
//<param name="e"> A reference of the event.
//<param name="nameFrame"> Name of the frame the button is part of.
//<param name="size"> Size of the circular progress bar.
public static void EventHandIn(RoutedEventArgs e, string nameFrame,
int size=100) {}

//Event raised when virtual hand leaves a YouButton
public static void EventHandOut(){}

//Timer that will raise the event of a click
//Example: YouHandTimer.autopress.Elapsed += new
//ElapsedEventHandler(Button_Click);
public static Timer autopress;

/* Window */

//Gets screens height
public static double getHeight(){}

//Gets screens width
public static double getWidth(){}
```

```
//Set a source of an image to a bitmap
//<param name="bitmap"> destination bitmap
//<param name="source"> source of image
//<param name="type"> type of the uri (Absolute, Relative,
//RelativeOrAbsolute)
public static void bitmapSource(BitmapImage bitmap,
string source, UriKind type)

//If the frame has ScrollArrow, this functions sizes and
//displays them correctly in the frame.
//<param name="ScrollLeft">Kinect Hover Button from the left side
//<param name="ScrollRight">Kinect Hover Button from the right
side
//<param name="h">Arrow Height
//<param name="w">Arrow Width
//<param name="lTop">Left Arrow Canvas Top Position
//<param name="lLeft">Left Arrow Canvas Left Position
//<param name="rTop">Right Arrow Canvas Top Position
//<param name="rLeft">Right Arrow Canvas Left Position
public static void setScrollArrows(KinectHoverButton ScrollLeft,
KinectHoverButton ScrollRight, double h, double w, double lTop,
double lLeft, double rTop, double rLeft){}

//If the frame has a UserViewer, this function sizes it and
//displays them correctly in the frame
//<param name="Viewer">Kinect User Viewer Control
//<param name="w">User Viewer Width
//<param name="h">User Viewer Height
//<param name="setBottom">User Viewer Set Bottom Canvas
//<param name="setRight">User Viewer Set Bottom Canvas
public static void setUserViewer(KinectUserViewer Viewer, double w,
double h, double setBottom, double setRight){}

/* Get XML and Paths files */

//Get a XML file with the App configuration
//<param name="plugin">A reference to the App's Plugin
public static XDocument getAppXml(YouPlugin plugin){}

//Get the YouInteract.exe directory path
public static string getYouInteractPath(){}
```

```
//Get the Log directory path
public static string getLogPath(){}

//Get the main configuration file
public static XDocument getPersonalConfigsPath(){}

//Get the background image chosen in the configuration
public static BitmapImage getTheme(){}
```

C.2 YouKinect

```
//Checks if Kinect is connected
//Returns: true if Kinect is connected; false if not connected
public static Boolean isKinectConnected(){}

//Skeleton Event that contains the Skeleton Array
//to be read by the developer
public static event SkeletonEventHandler
SkeletonEvent = delegate {};

//InteractionStream Event that contains the UserInfo Array
//to be read by the developer
public static event InteractionEventHandler
InteractionEvent = delegate {};

//DEBUG ONLY
//Stop connection with Kinect
public static void StopKinect(){}

//DEBUG ONLY
//Initializes connection with Kinect
public static void initKinect(){}

//DEBUG ONLY
//Binds a frame Kinect Region with Kinect sensor
public static void bindRegion(KinectRegion region){}

//DEBUG ONLY
//Unbinds a frame Kinect Region from Kinect sensor
public static void unbindRegion(KinectRegion region){}
```

C.3 YouNavigation

```
//Developer requests a change to a different frame
//from the same application.
//<param name="sourcePage">Current page
//<param name="destinationName">Name of the destination page
public static void requestFrameChange(YouPlugin sourcePage,
String destinationName){}

//Developer requests a change to the main menu
//<param name="sourcePage">Current page
public static void navigateToMainMenu(YouPlugin sourcePage){}

/* Interface that needs to be implemented in every page */
public interface YouPlugin
{
    //A function that returns the Page's Name
    String getName();

    //A function that returns the name of the App
    String getAppName();

    //A function that returns an instance of the page
    Page getPage();

    //A function to identify a page's Kinect Requirements
    KinectRequirements getKinectRequirements();

    //A function that returns an instance
    //of the Kinect Region
    KinectRegion getRegion();

    //A function that returns whether the page is the
    //starting page the application
    bool getIsFirstPage();
}
```

C.4 YouControls

```
//A button that continually triggers a click when
//virtual hand hovers over it
public class KinectHoverButton:KinectButtonBase{}

//A button that responds to a Timer and Push click
//with the virtual hand
public class YouButton:KinectTileButton{}
```

C.5 YouLog

```
//Creates a net .json with the name of the application
//<param name="app">Application's name
public static void  CreateJson(string app) {}

//Inserts a new property in the app .json with a
//specific value
//<param name="app">Application's name
//<param name="prop_name">Property's name
//<param name="value">Value of that property
public static void InsertProp(string app,
string prop_name, string value) {}

//Inserts a new array property in the app .json with a
//specific value
//<param name="app">Application's name
//<param name="prop_name">Property's name
//<param name="value">Value of that property
public static void InsertPropArray(string app,
string prop_name, string value) {}

//Updates value of a property
//<param name="app">Application's name
//<param name="prop_name">Property's name
//<param name="value">Value of that property
public static void UpdateProp(string app, string prop_name,
string value) {}

//Updates value of a property array
//<param name="app">Application's name
//<param name="prop_name">Property's name
//<param name="value">Value of that property
public static void UpdatePropArray(string app, string prop_name,
string value) {}
```

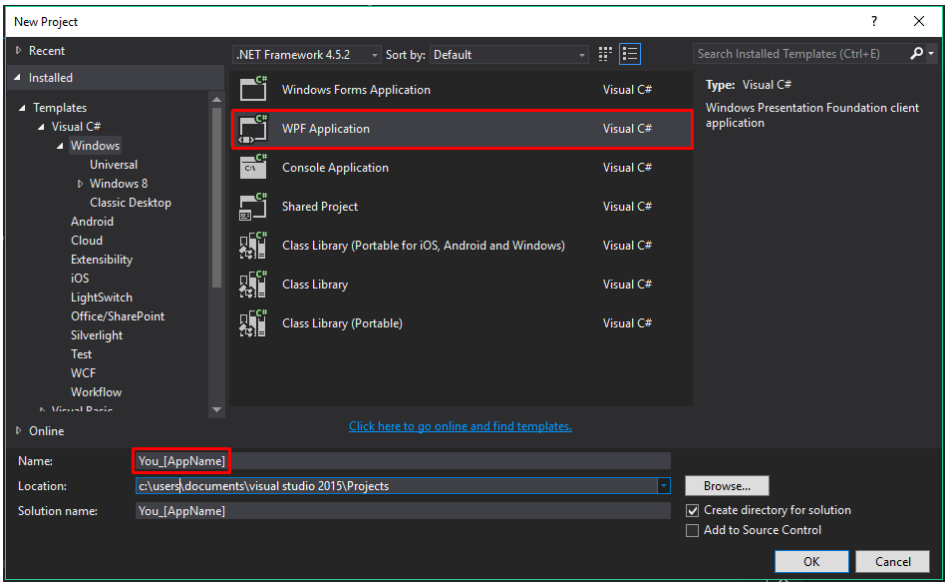
Appendix D

YouInteract WPF application development tutorial

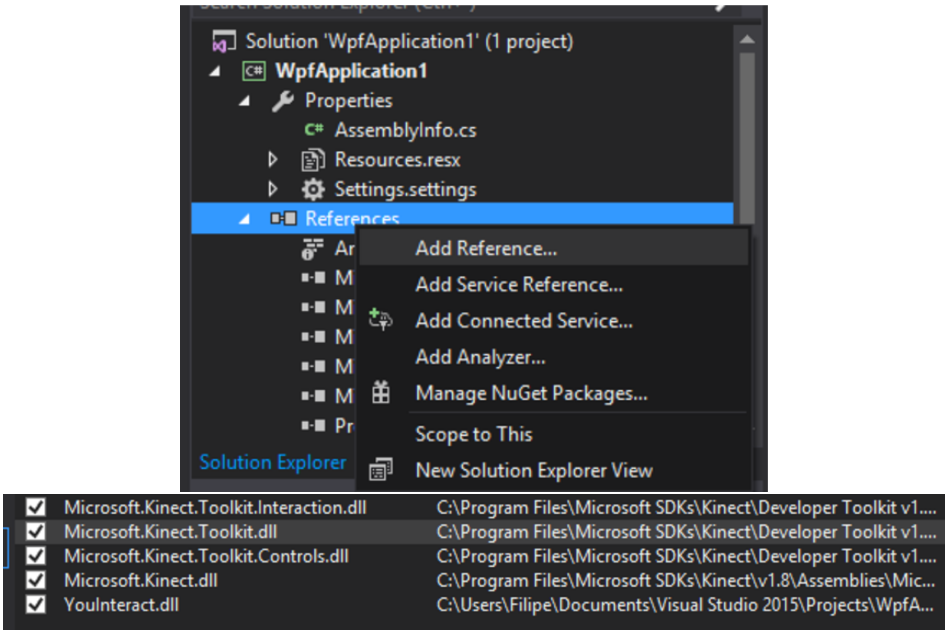
In order to develop an application for YouInteract, firstly ensure the computer has .NET Framework and Microsoft Kinect SDK 1.8 installed (Appendix F).

After check the software described above is installed, please follow the next steps:

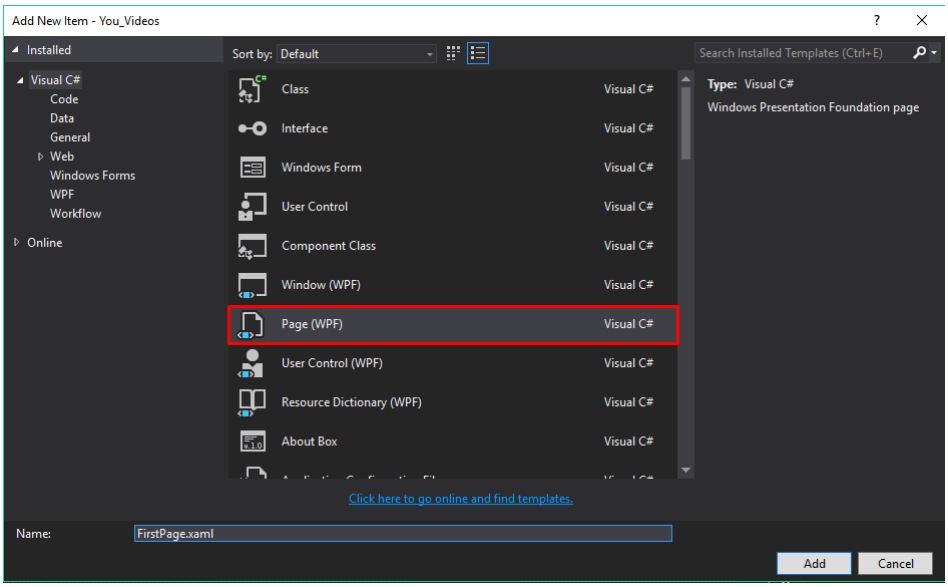
1. Start a new WPF project. The name MUST start with "You_[AppName]".



2. Add references to Kinect SDK .dlls and to YouInteract API .dll



3. Delete the Window files that WPF inserts by default and add a new page (the first page of the application).



4. Add the obligatory Plugin to the page and implement its methods (Appendix C).

```
using YouInteract.YouBase;
using YouInteract.YouSystem;

namespace WpfApplication1
{
    /// <summary>
    /// Interaction logic for ex1.xaml
    /// </summary>
    4 references
    public partial class ex1 : Page, YouPlugin
    {
```

5. At the page XAML add the references to Kinect and YouInteract API.
   xmlns:k=”http://schemas.microsoft.com/kinect/2013”
   xmlns:u=”clr-namespace:YouInteract.YouAPI;assembly=YouInteract”

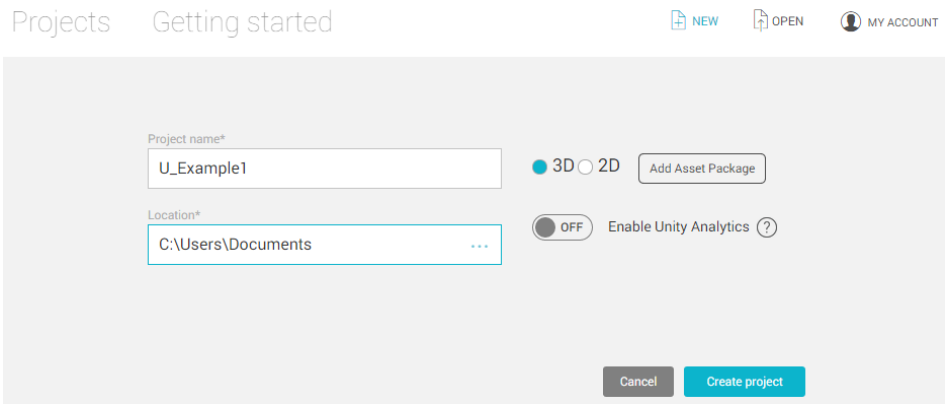
```
<Page x:Class="WpfApplication1.ex1"
      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
      xmlns:local="clr-namespace:WpfApplication1"
      xmlns:k="http://schemas.microsoft.com/kinect/2013"
      xmlns:u="clr-namespace:YouInteract.YouAPI;assembly=YouInteract"
```

Appendix E

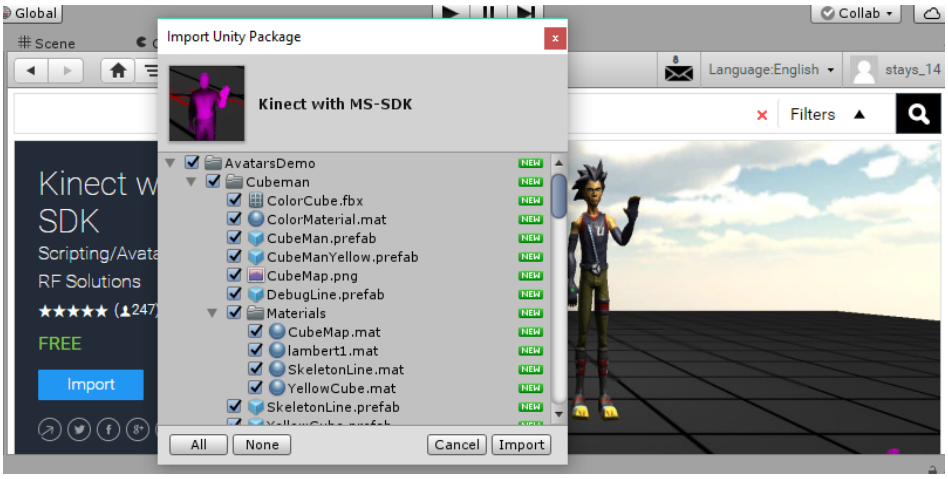
YouInteract Unity application development tutorial

A Unity application developed for YouInteract must support gesture interaction so, integrating Microsoft Kinect is fundamental. Then it is also fundamental to have a way of going back to YouInteract's menu so a button can be inserted that when is clicked, the application closes and YouInteract's menu is shown.

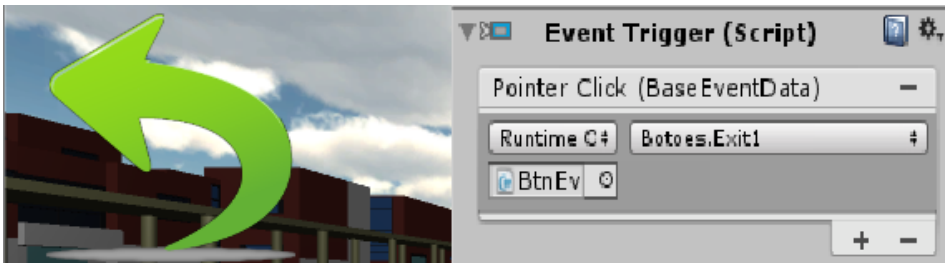
1. Start a new Unity 3D application. Its name MUST be "U_[name of application]".



2. Import the asset responsible for integrating Microsoft Kinect.
   (a) Go to Unity store.
   (b) Search for "Kinect with MS-SDK". It is free and it was developed by RF Solutions.
   (c) Import the asset.



3. Add a button that when clicked, the applications closes.



Appendix F

Set up a YouInteract device

In order to have YouInteract system fully installed in the computer, it is necessary to make the following steps:

1. Kinect requirements - https://msdn.microsoft.com/en-us/library/hh855359.aspx

   Firstly it is important to check if the computer has hardware good enough to integrate Microsoft Kinect.

2. Install a configure Dropbox on the computer - https://www.dropbox.com

   After installing it is necessary to login with an appropriate account with YouInteract. The details about the account must be asked to the project manager.

3. Install Windows .NET Framework 4.6.* - https://www.microsoft.com/net/download/framework

   In order to run or even develop for YouInteract, it is needed at least the 4.6.1 version of .NET Framework.

4. Install Kinect SDK 1.8 and connect Kinect to the computer - https://www.microsoft.com/en-us/download/details.aspx?id=40278

   Download and install the SDK providen by Microsoft. In order to assure the equipment is fully working, please test some of the samples that come with the SDK.

5. Access dropbox files and run the executable YouInteract.exe.